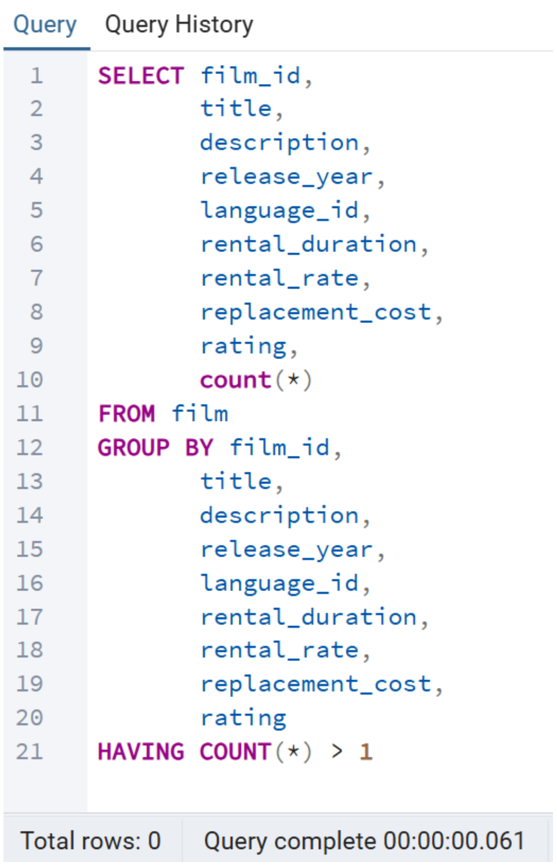
Summarizing & Cleaning Data in SQL

1. Check for and clean dirty data: Find out if the film table and the customer table contain any dirty data, specifically non-uniform or duplicate data, or missing values. Create a new "Answers 3.6" document and copy-paste your queries into it. Next to each query write 2 to 3 sentences explaining how you would clean the data (even if the data is not dirty).

Film table / Duplicate Data



```
SELECT film_id,
       title,
       description,
       release_year,
       language_id,
       rental_duration,
       rental_rate,
       replacement_cost,
       rating,
       count(*)
FROM film
GROUP BY film_id,
       title,
       description,
       release_year,
       language_id,
       rental_duration,
       rental_rate,
       replacement_cost,
       rating
HAVING COUNT(*) > 1
```

```
Total rows: 0    Query complete 00:00:00.061
```

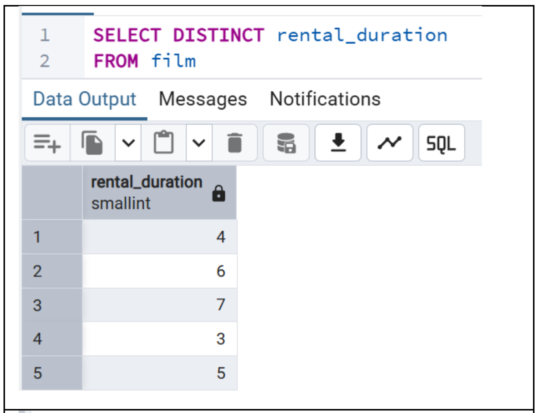
No duplicate data was found in the film table.

If there is duplicate data,

(1) use DISTINCT or GROUP BY in queries to return unique records for analysis.

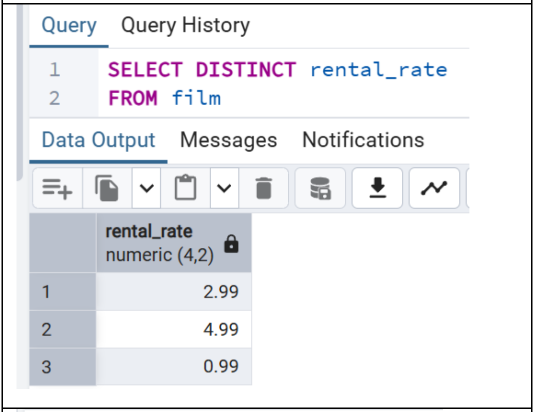
(2) create a view to create unique records if permitted by supervisor.

Film table / Non-Uniform Data



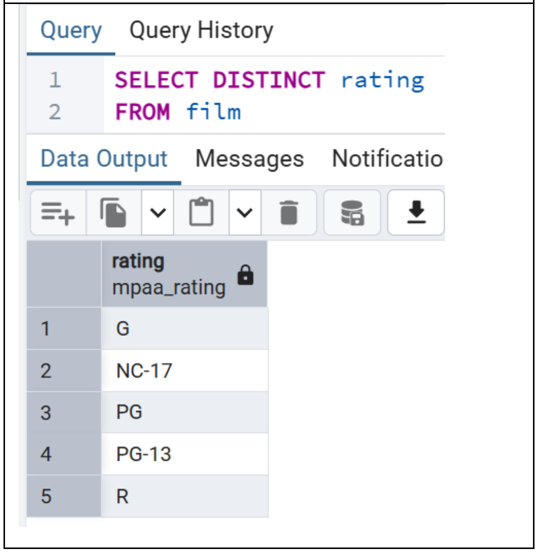
```sql
SELECT DISTINCT rental_duration
FROM film
```

	rental_duration smallint
1	4
2	6
3	7
4	3
5	5



```sql
SELECT DISTINCT rental_rate
FROM film
```

	rental_rate numeric (4,2)
1	2.99
2	4.99
3	0.99



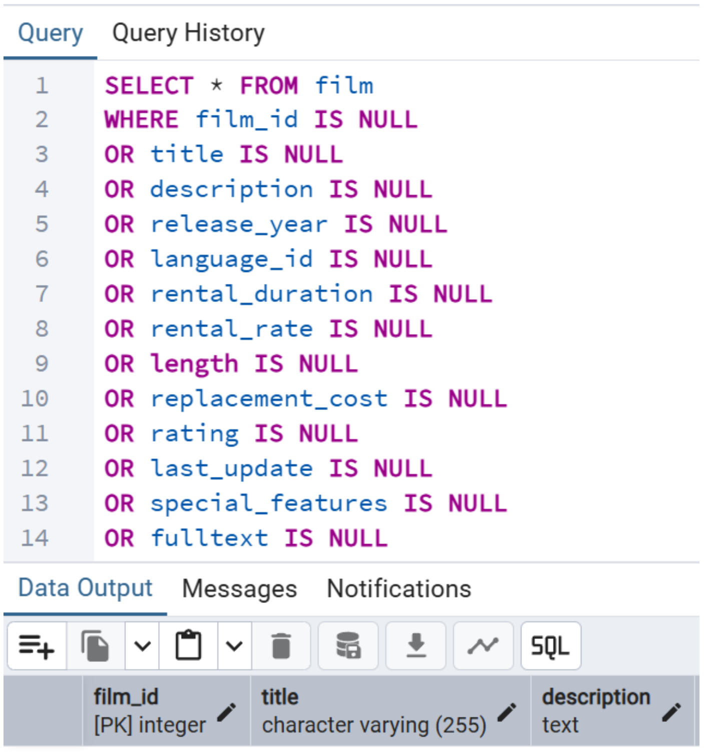
```sql
SELECT DISTINCT rating
FROM film
```

	rating mpaa_rating
1	G
2	NC-17
3	PG
4	PG-13
5	R

No non-uniform data was found.
If there is non-uniform data, I will use UPDATE the table using the commands as follows:

UPDATE film
SET column_name = ‘value’
Where column_name IN (‘value1’, ‘value2’, ‘value3’)

Film table/Missing Values



```sql
SELECT * FROM film
WHERE film_id IS NULL
OR title IS NULL
OR description IS NULL
OR release_year IS NULL
OR language_id IS NULL
OR rental_duration IS NULL
OR rental_rate IS NULL
OR length IS NULL
OR replacement_cost IS NULL
OR rating IS NULL
OR last_update IS NULL
OR special_features IS NULL
OR fulltext IS NULL
```

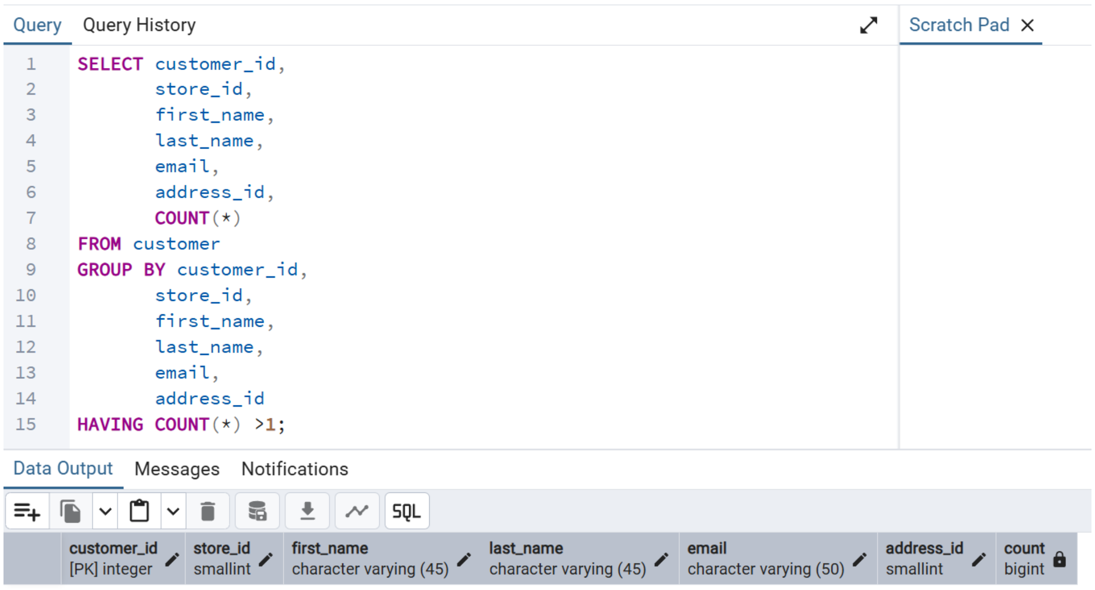
film_id [PK] integer	title character varying (255)	description text

No missing values were identified in the table. If missing values are present in future analyses and account for less than 5% of the data, they will be addressed using either listwise deletion or mean imputation. If missing values exceed 5%, I will ignore them and clearly explained why I do that.

The SQL commands for mean imputations are as follows:

UPDATE film
SET = AVG(column_name)
WHERE column_name IS NULL

Customer table / Duplicate Data



```sql
SELECT customer_id,
       store_id,
       first_name,
       last_name,
       email,
       address_id,
       COUNT(*)
FROM customer
GROUP BY customer_id,
       store_id,
       first_name,
       last_name,
       email,
       address_id
HAVING COUNT(*) >1;
```

customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	count bigint

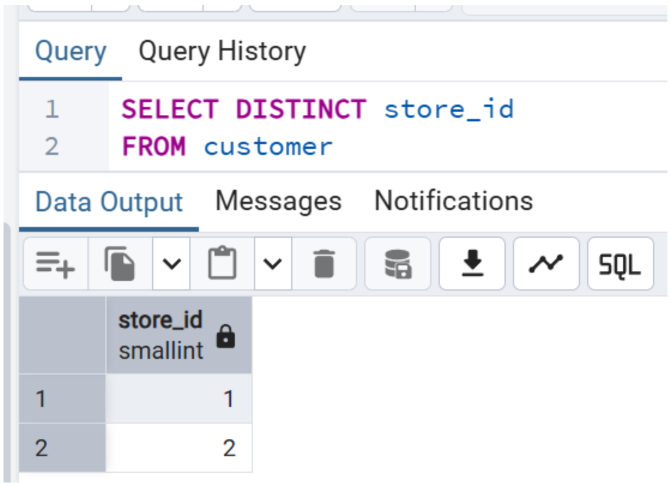
No duplicate data was found in the customer table.

If there is duplicate data,

(1) use DISTINCT or GROUP BY in queries to return unique records for analysis.

(2) create a view to create unique records if permitted by supervisor.

Customer table / Non-Uniform Data



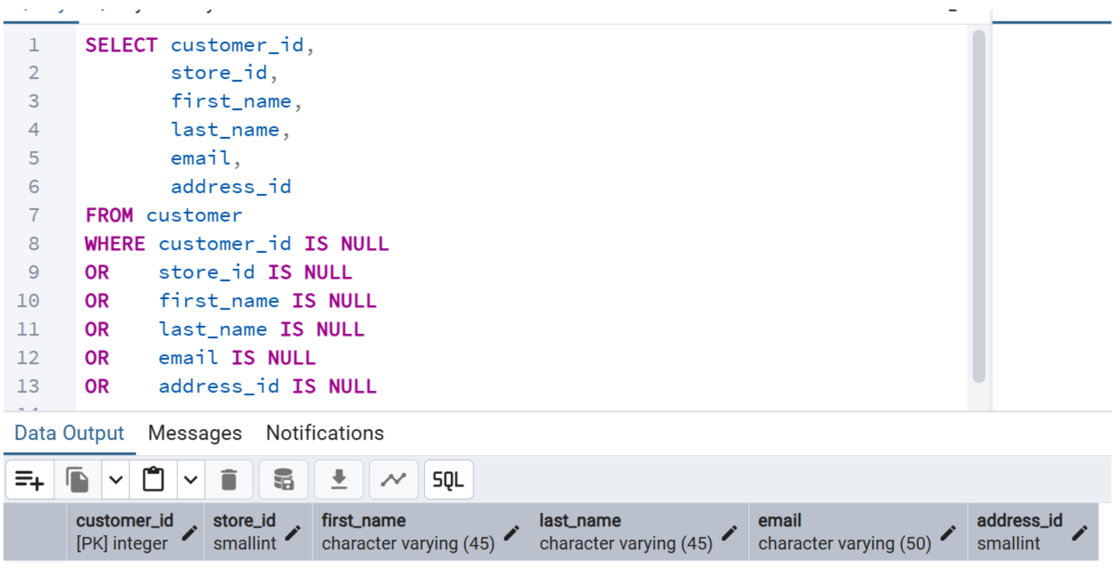
```sql
SELECT DISTINCT store_id
FROM customer
```

	store_id smallint
1	1
2	2

No non-uniform data was found.
If there is non-uniform data, I will use UPDATE the table using the commands as follows:

UPDATE customer
SET column_name = ‘value’
Where column_name IN (‘value1’, ‘value2’, ‘value3’)

Customer table/Missing Values


```
SELECT customer_id,
       store_id,
       first_name,
       last_name,
       email,
       address_id
FROM customer
WHERE customer_id IS NULL
OR    store_id IS NULL
OR    first_name IS NULL
OR    last_name IS NULL
OR    email IS NULL
OR    address_id IS NULL
```

customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint

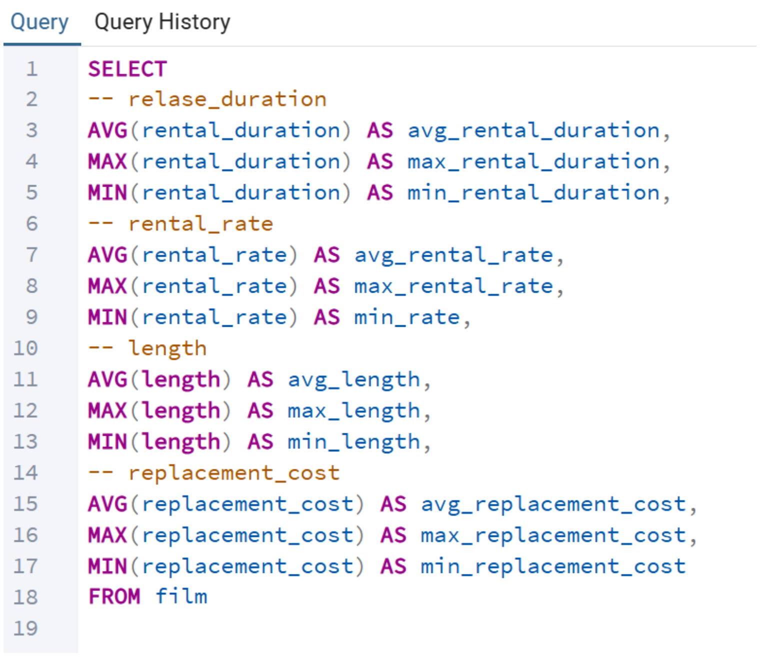
No missing values were identified in the table. If missing values are present in future analyses and account for less than 5% of the data, they will be addressed using either listwise deletion or mean imputation. If missing values exceed 5%, I will ignore them and clearly explained why I do that.

The SQL commands for mean imputations are as follows:

UPDATE customer
SET = AVG(column_name)
WHERE column_name IS NULL

2. **Summarize your data: Use SQL to calculate descriptive statistics for both the film table and the customer table. For numerical columns, this means finding the minimum, maximum, and average values. For non-numerical columns, calculate the mode value. Copy-paste your SQL queries and their outputs into your answers document.**

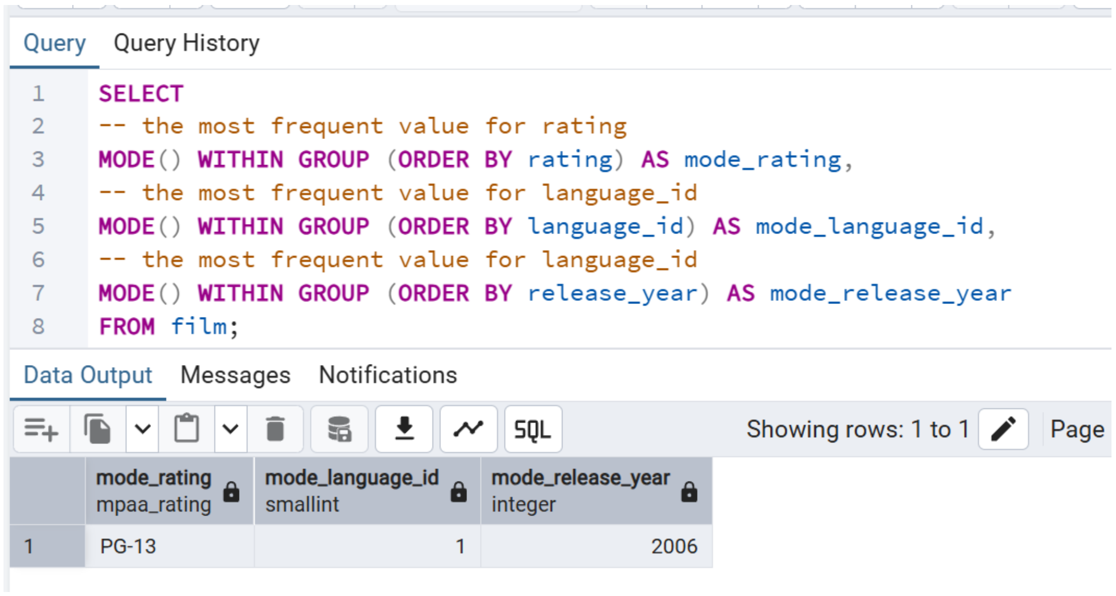
Film Table



```sql
SELECT
-- relase_duration
AVG(rental_duration) AS avg_rental_duration,
MAX(rental_duration) AS max_rental_duration,
MIN(rental_duration) AS min_rental_duration,
-- rental_rate
AVG(rental_rate) AS avg_rental_rate,
MAX(rental_rate) AS max_rental_rate,
MIN(rental_rate) AS min_rate,
-- length
AVG(length) AS avg_length,
MAX(length) AS max_length,
MIN(length) AS min_length,
-- replacement_cost
AVG(replacement_cost) AS avg_replacement_cost,
MAX(replacement_cost) AS max_replacement_cost,
MIN(replacement_cost) AS min_replacement_cost
FROM film

```

Column	Average	Maximum	Minimum
rental_duration	4.99	7	3
rental_rate	2.98	4.99	0.99
length	115.272	185	46
replacement_cost	19.984	29.99	9.99

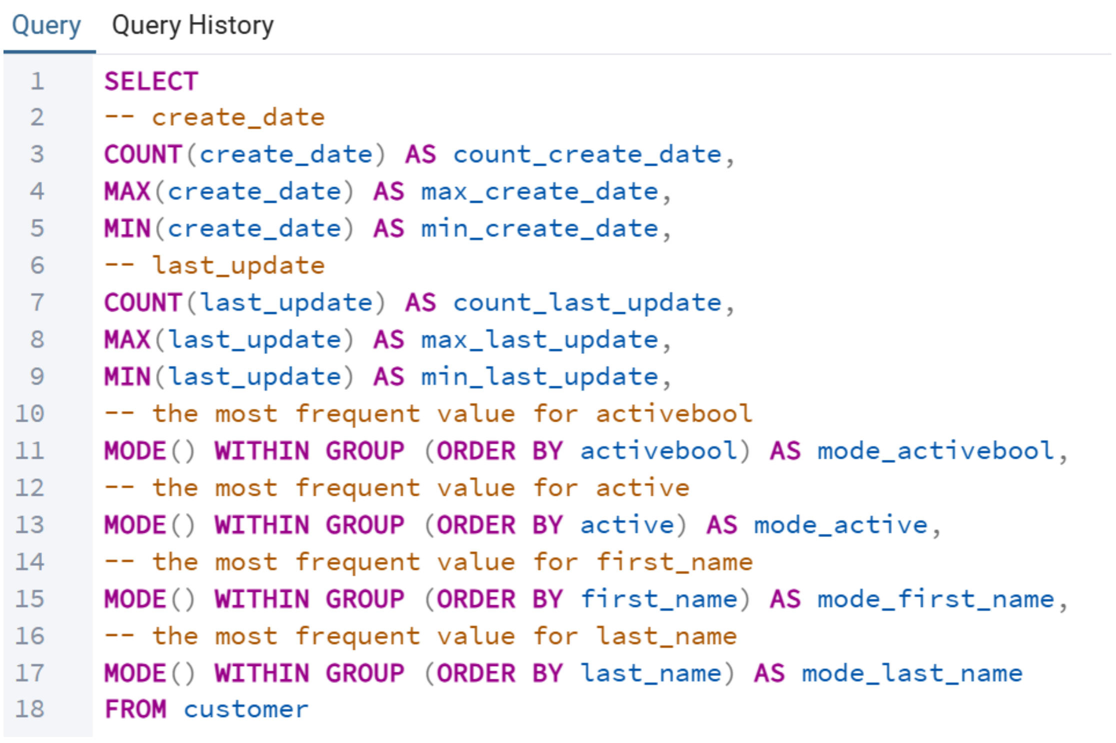


```sql
SELECT
-- the most frequent value for rating
MODE() WITHIN GROUP (ORDER BY rating) AS mode_rating,
-- the most frequent value for language_id
MODE() WITHIN GROUP (ORDER BY language_id) AS mode_language_id,
-- the most frequent value for language_id
MODE() WITHIN GROUP (ORDER BY release_year) AS mode_release_year
FROM film;
```

	mode_rating mpaa_rating	mode_language_id smallint	mode_release_year integer
1	PG-13	1	2006

Column	Mode
language_id	1
rating	PG- 13
release_year	2006

Customer Table



```sql
SELECT
-- create_date
COUNT(create_date) AS count_create_date,
MAX(create_date) AS max_create_date,
MIN(create_date) AS min_create_date,
-- last_update
COUNT(last_update) AS count_last_update,
MAX(last_update) AS max_last_update,
MIN(last_update) AS min_last_update,
-- the most frequent value for activebool
MODE() WITHIN GROUP (ORDER BY activebool) AS mode_activebool,
-- the most frequent value for active
MODE() WITHIN GROUP (ORDER BY active) AS mode_active,
-- the most frequent value for first_name
MODE() WITHIN GROUP (ORDER BY first_name) AS mode_first_name,
-- the most frequent value for last_name
MODE() WITHIN GROUP (ORDER BY last_name) AS mode_last_name
FROM customer
```

Column	Count	Maximum	Minimum
creation_date	599	2006-02-14	2006-02-14
last_update	599	2013-05-26 14:49:45.738	2013-05-26 14:49:45.738

Column	Mode
activebool	true
active	1
first_name	Jamie
Last_name	Abney

3. **Reflect on your work: Back in Achievement 1 you learned about data profiling in Excel. Based on your previous experience, which tool (Excel or SQL) do you think is more effective for data profiling, and why? Consider their respective functions, ease of use, and speed. Write a short paragraph in the running document that you have started.**

<u>Ease of use</u>: Excel is easier for starters. SQL might get easier after you get familiar with the commands and rules.

<u>Functions</u>: SQL is more powerful because it can deal with complicated tasks with several lines of commands. In comparison, Excel is much less automatic.

<u>Speed</u>: SQL is faster than Excel especially when dealing with large datasets.