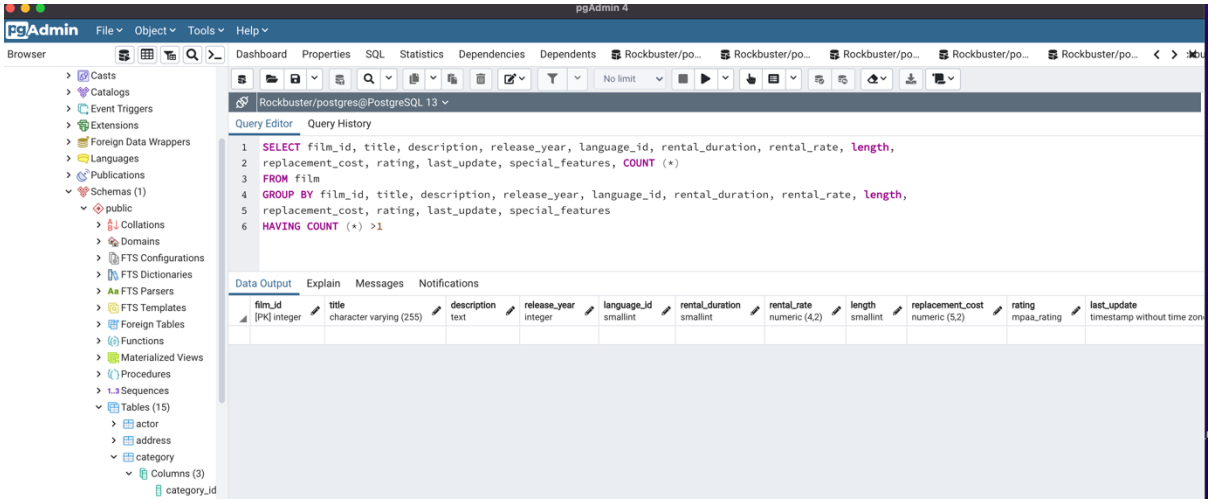
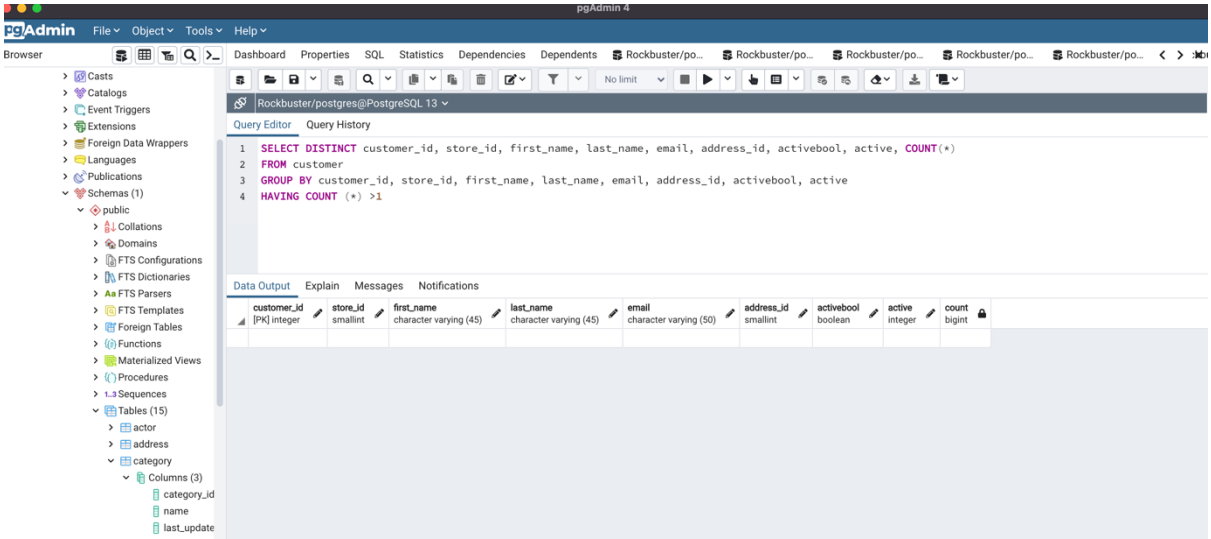
TASK_6: Summarizing & Cleaning Data in SQL

1. Check for and clean dirty data:

Duplicate Data from Film

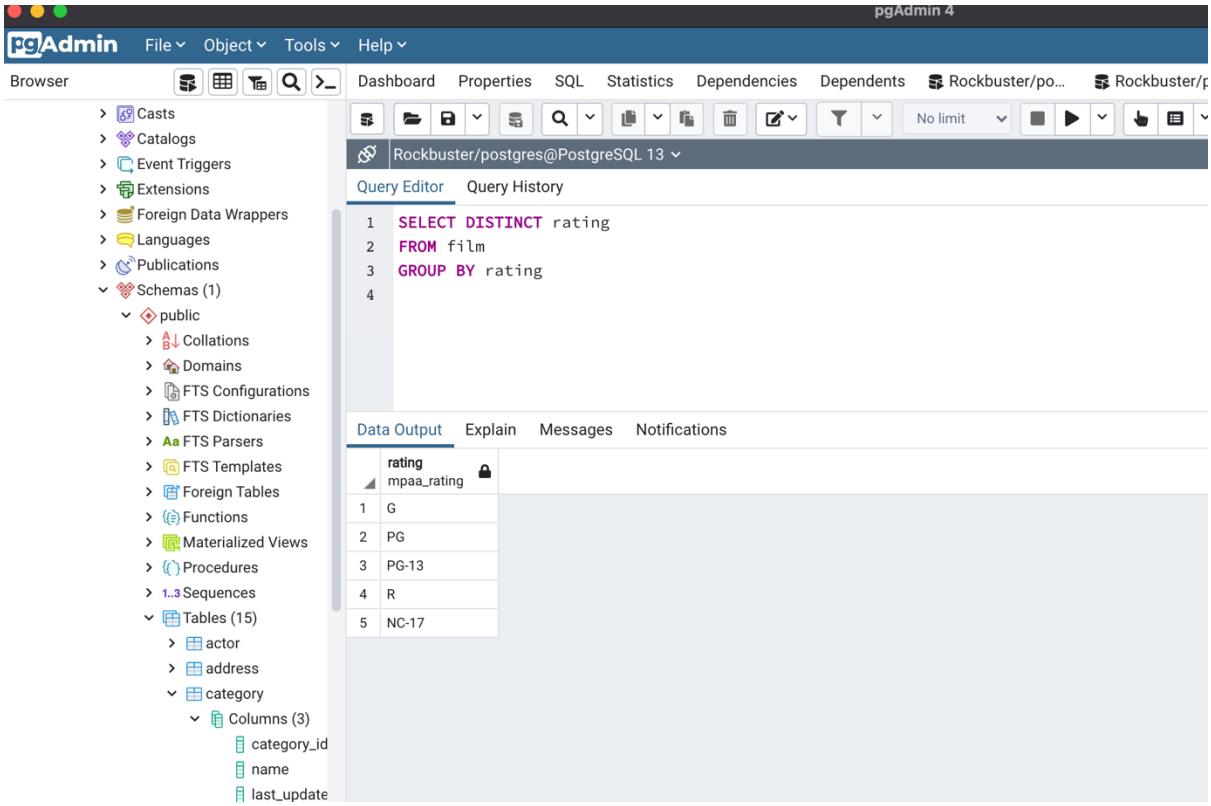


Duplicate Data from Customer



In both cases there is no duplicate data.

Non-Uniform Data



Based on our results, there is also none of non-uniform values. If there would be, we can use the UPDATE command in combination with SET and WHERE.

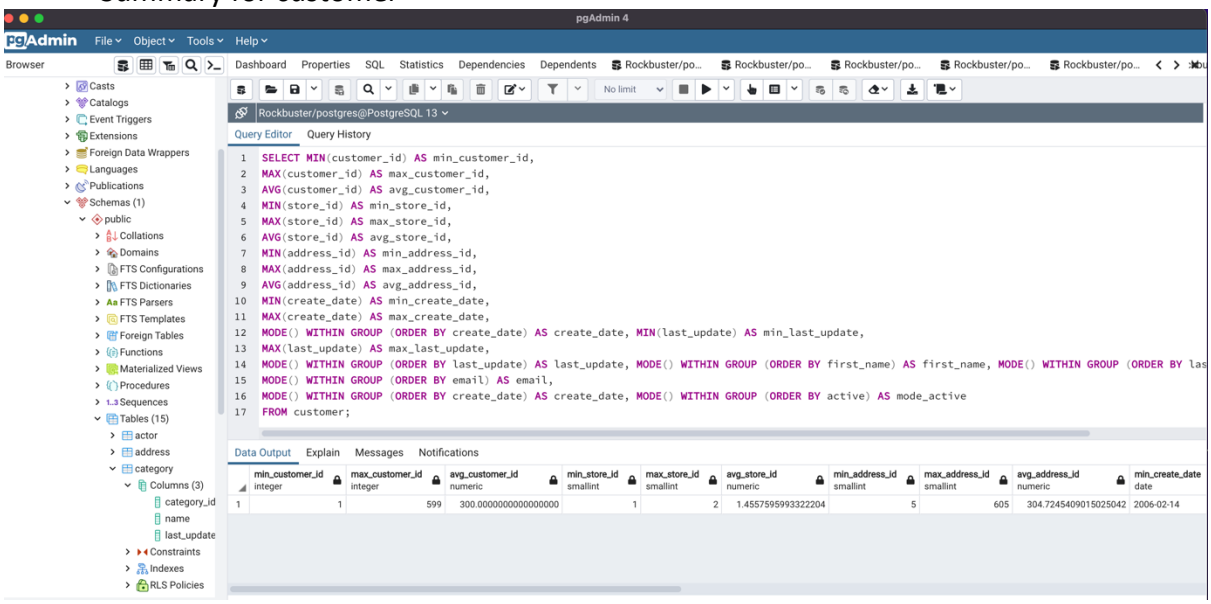
For Incorrect data, I would discuss with my team to find a solution, because in this case instead of writing commands, logical and critical thinking is more important.

To find missing information, firstly I would visualize the data. After detecting the missing value, we might replace them. As example:

UPDATE tablename
SET = AVG(coll)
WHERE coll IS NULL

2. Summarize your data

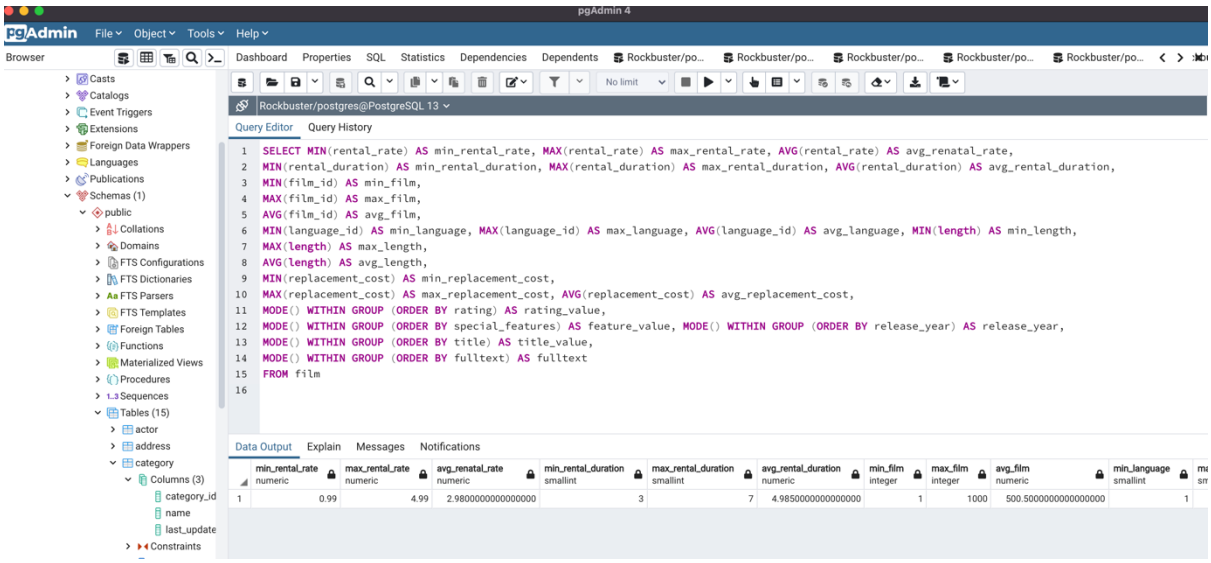
Summary for customer



--descriptive statistics for customer table
SELECT MIN(customer_id) AS min_customer_id, MAX(customer_id) AS max_customer_id,
AVG(customer_id) AS avg_customer_id,
MIN(store_id) AS min_store_id,
MAX(store_id) AS max_store_id,
AVG(store_id) AS avg_store_id,
MIN(address_id) AS min_address_id,
MAX(address_id) AS max_address_id,
AVG(address_id) AS avg_address_id,
MIN(create_date) AS min_create_date,
MAX(create_date) AS max_create_date,
MODE() WITHIN GROUP (ORDER BY create_date) AS create_date, MIN(last_update) AS min_last_update, MAX(last_update) AS max_last_update,
MODE() WITHIN GROUP (ORDER BY last_update) AS last_update, MODE() WITHIN GROUP (ORDER BY first_name) AS first_name, MODE() WITHIN GROUP (ORDER BY last_name) AS last_name, MODE() WITHIN GROUP (ORDER BY email) AS email,
MODE() WITHIN GROUP (ORDER BY create_date) AS create_date, MODE() WITHIN GROUP (ORDER BY active) AS mode_active

FROM customer;

Summary for Film



--descriptive statistics for film table
SELECT MIN(rental_rate) AS min_renatl_rate, MAX(rental_rate) AS max_rental_rate, AVG(rental_rate) AS avg_renatal_rate, MIN(rental_duration) AS min_rental_duration, MAX(rental_duration) AS max_rental_duration, AVG(rental_duration) AS avg_rental_duration, MIN(film_id) AS min_film,
MAX(film_id) AS max_film,
AVG(film_id) AS avg_film,
MIN(language_id) AS min_language, MAX(language_id) AS max_language, AVG(language_id) AS avg_language, MIN(length) AS min_length, MAX(length) AS max_length,
AVG(length) AS avg_length,
MIN(replacement_cost) AS min_replacement_cost,
MAX(replacement_cost) AS max_replacement_cost, AVG(replacement_cost) AS avg_replacement_cost,
MODE() WITHIN GROUP (ORDER BY rating) AS rating_value,
MODE() WITHIN GROUP (ORDER BY special_features) AS feature_value, MODE() WITHIN GROUP (ORDER BY release_year) AS release_year, MODE() WITHIN GROUP (ORDER BY title) AS title_value,
MODE() WITHIN GROUP (ORDER BY fulltext) AS fulltext
FROM film

3. Excel is useful for quick visualizations and summaries of data, whereas SQL is necessary for working with large volumes of data, managing databases, and using relational databases to their full potential. Excel gets slower the more data you ask it to handle. Excel cannot store more than one million lines of data. SQL is fast and can handle large loads of data. Unlike Excel, SQL can handle well over one million fields of data with ease. SQL queries are also more flexible and powerful.