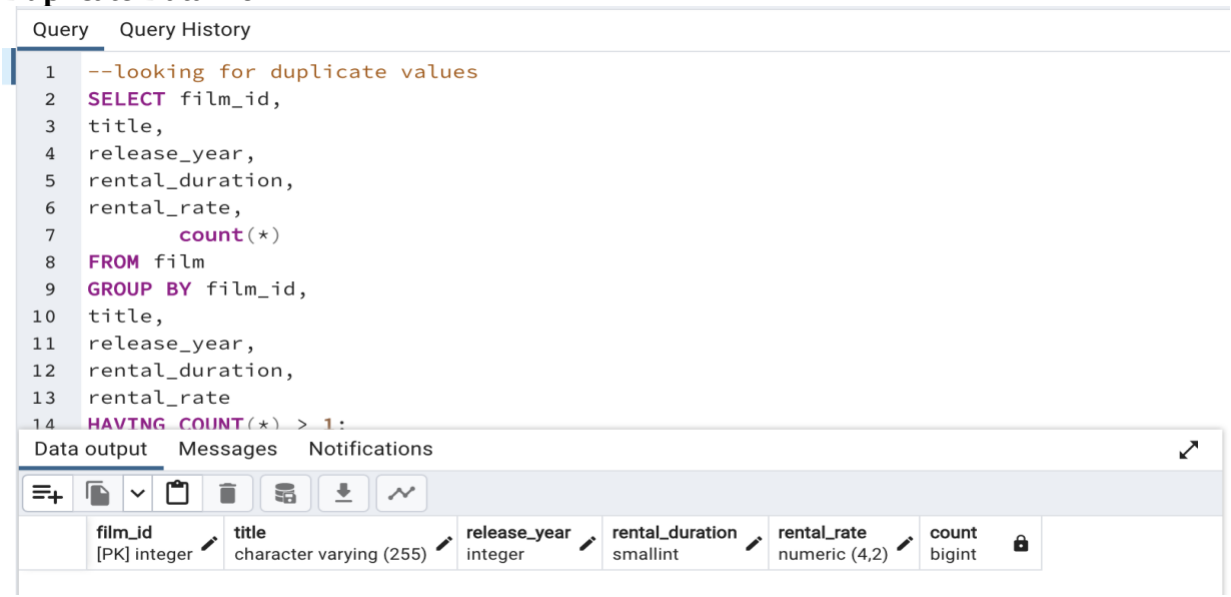
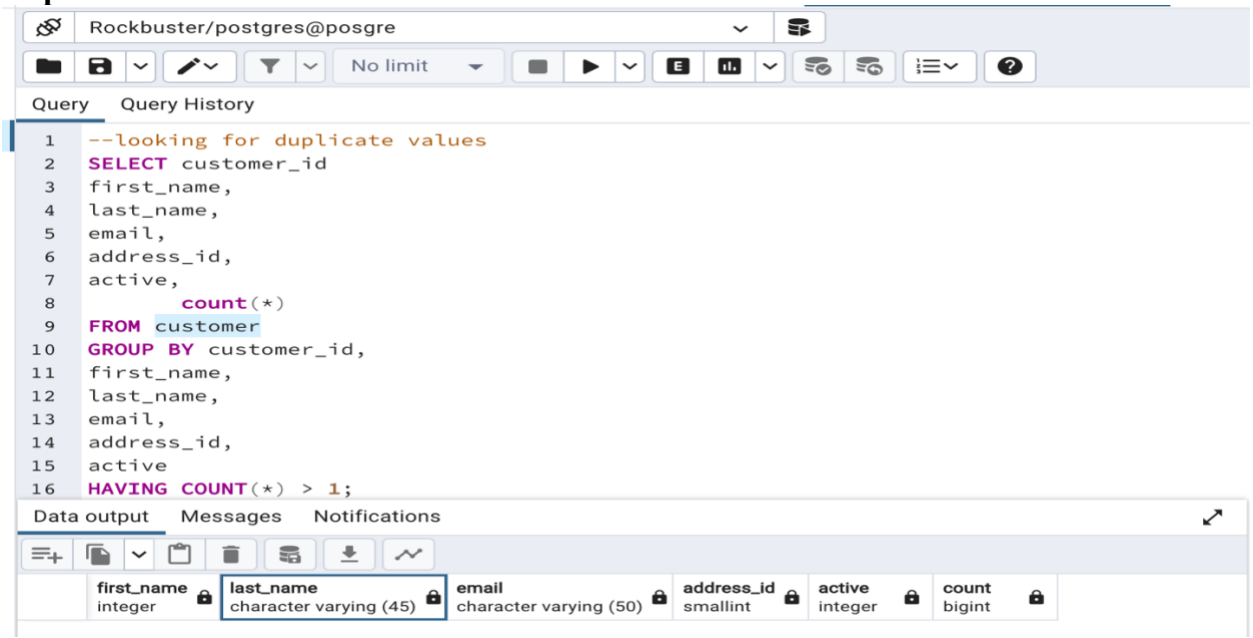
3.6: Summarizing & Cleaning Data in SQL

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).

Duplicate Data from Film



Duplicate Data from Customer



There is no returned duplicate value. There are two ways of dealing with duplicate value if you have permission to alter the database. • Create a virtual table "View" where unique records can be

selected • Delete duplicate record from the table or View However, if altering table is not permiQed, we can use GROUP BY or DISTINCT to select unique records

Non-Uniform Data

Query | Query History

```
1  --looking for Non-Uniform Data
2  SELECT DISTINCT rating
3  FROM film
4  GROUP BY rating
```

Data output | Messages | Notifications

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

Values are homogeneous, but in order to fix it on the table if there was an issue, I would use the UPDATE command combined with SET and WHERE, to replace the values that should be differently represented.

Incorrect Data

For incorrect data, logic and critical thinking can be used instead of commands, as you can analyse the tables and data for outliers or things that are wrongly displayed.

A mix of the commands show before can also help, by grouping and organizing the information in a more visible way in order for the analyst to review it.

Missing Data

This can also be fixed with logic, as you can use commands to better visualize the data and find missing or null information.

In some cases where a column has too much information missing, it might even be valid to remove it from queries.

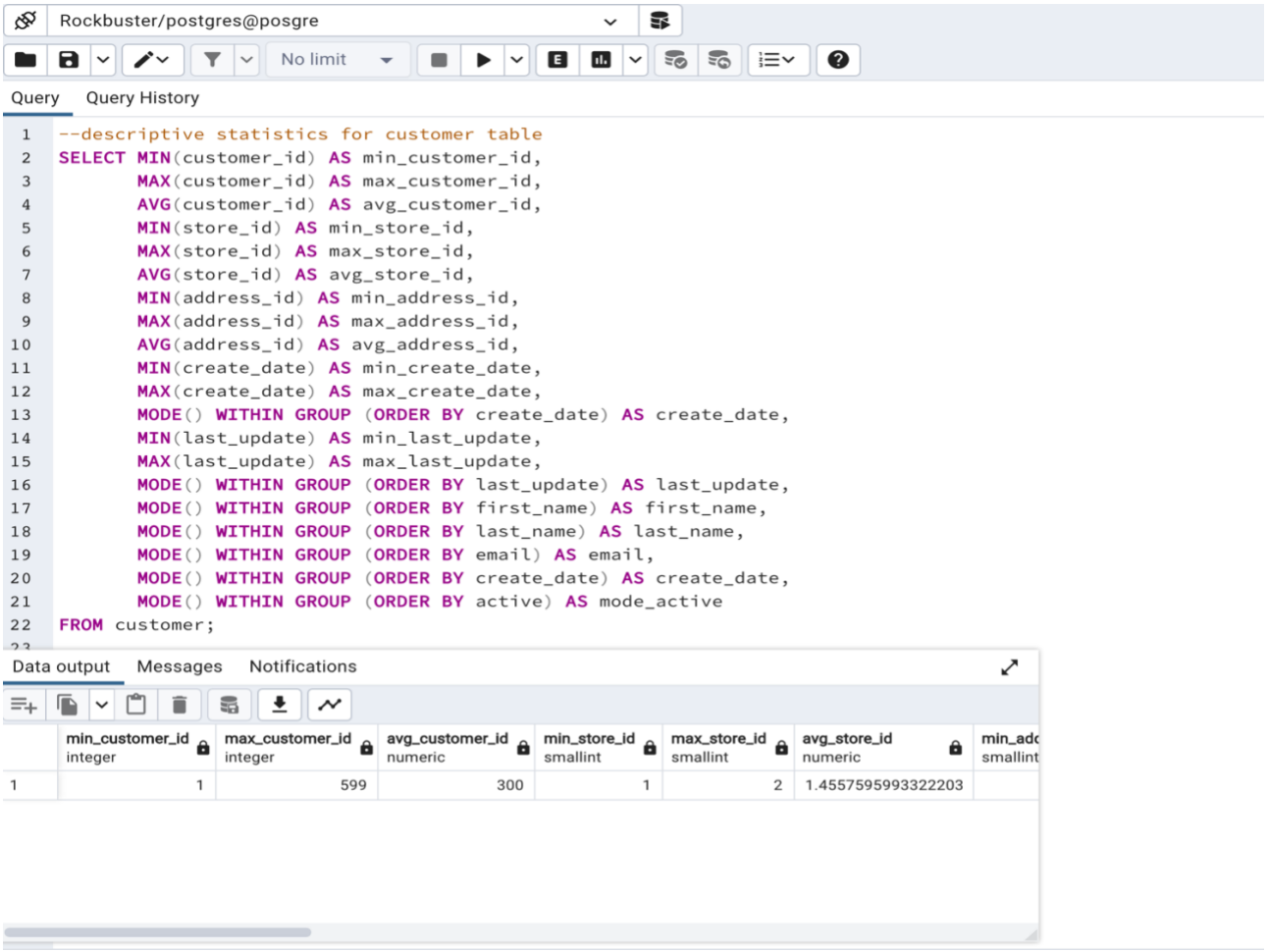
A solution for it after finding the missing data, is to replace it with the average of the remaining informed data (if appropriate). The following command can assist with that:

UPDATE tablename
SET = AVG(col1)
WHERE col1 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.

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;
```

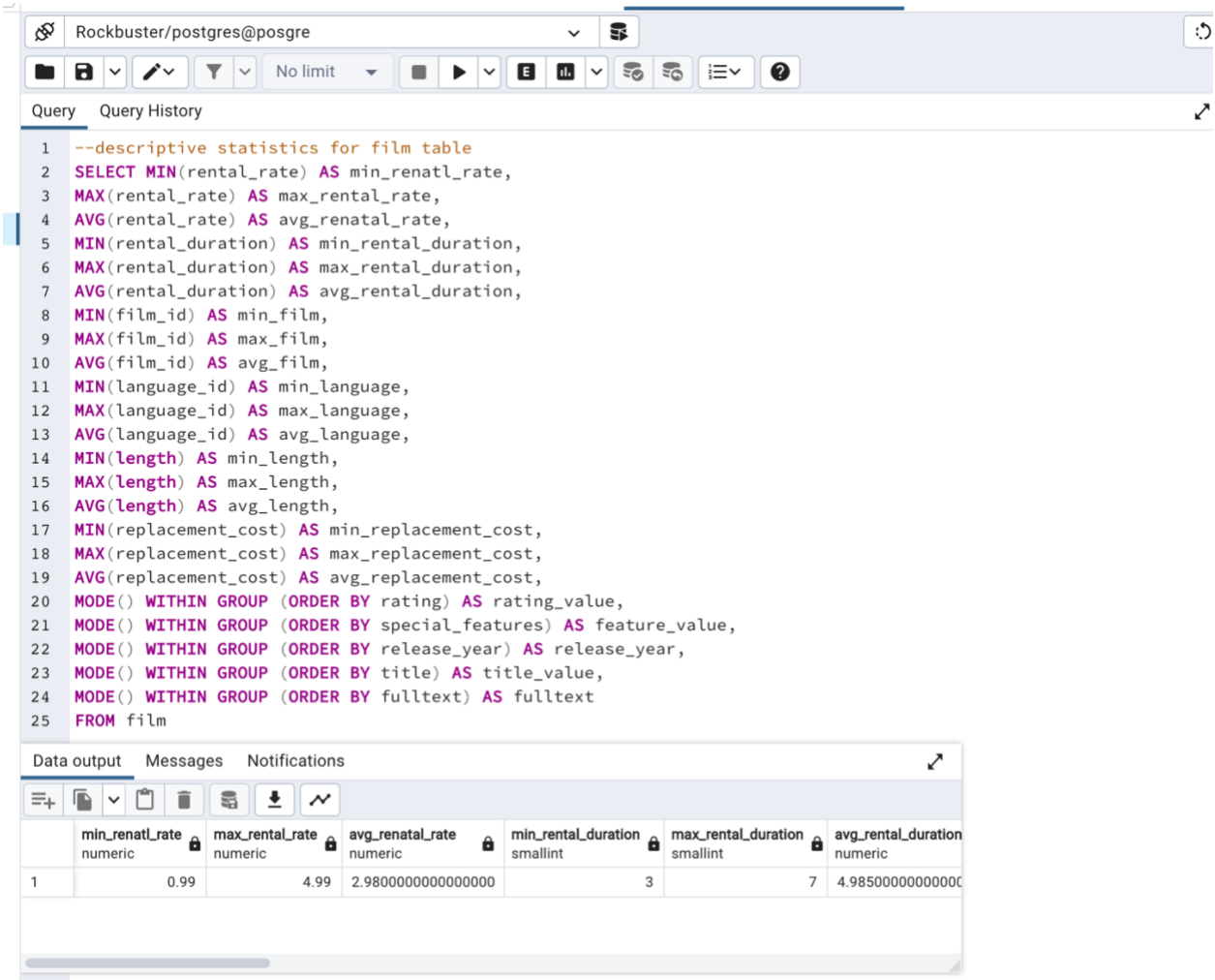


For some information instead of MAX, MIN and AVG I used MAX, MIN and MODE, it makes more sense to see when something happened more often for dates rather than the average date.

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



For some information instead of MAX, MIN and AVG I used MAX, MIN and MODE, it makes more sense to see when something happened more often for dates rather than the average date.

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.

Excel works great with a small amount of data. At the same time, it will be easy to view data using pivot tables. However, renaming the output (alias) for an aggregation column will take longer, it's easy to work with huge data in SQL. Using SQL data profiling is getting easier and faster. The specific result/request for details will be returned immediately with the correct syntax.
It might be nice to use Excel for data cleansing and processing small datasets, but then for working with it or when processing huge amounts of data, I would choose SQL.