Task 3.6 Summarising & Cleaning Data in SQL

By Lee Heng Chuah

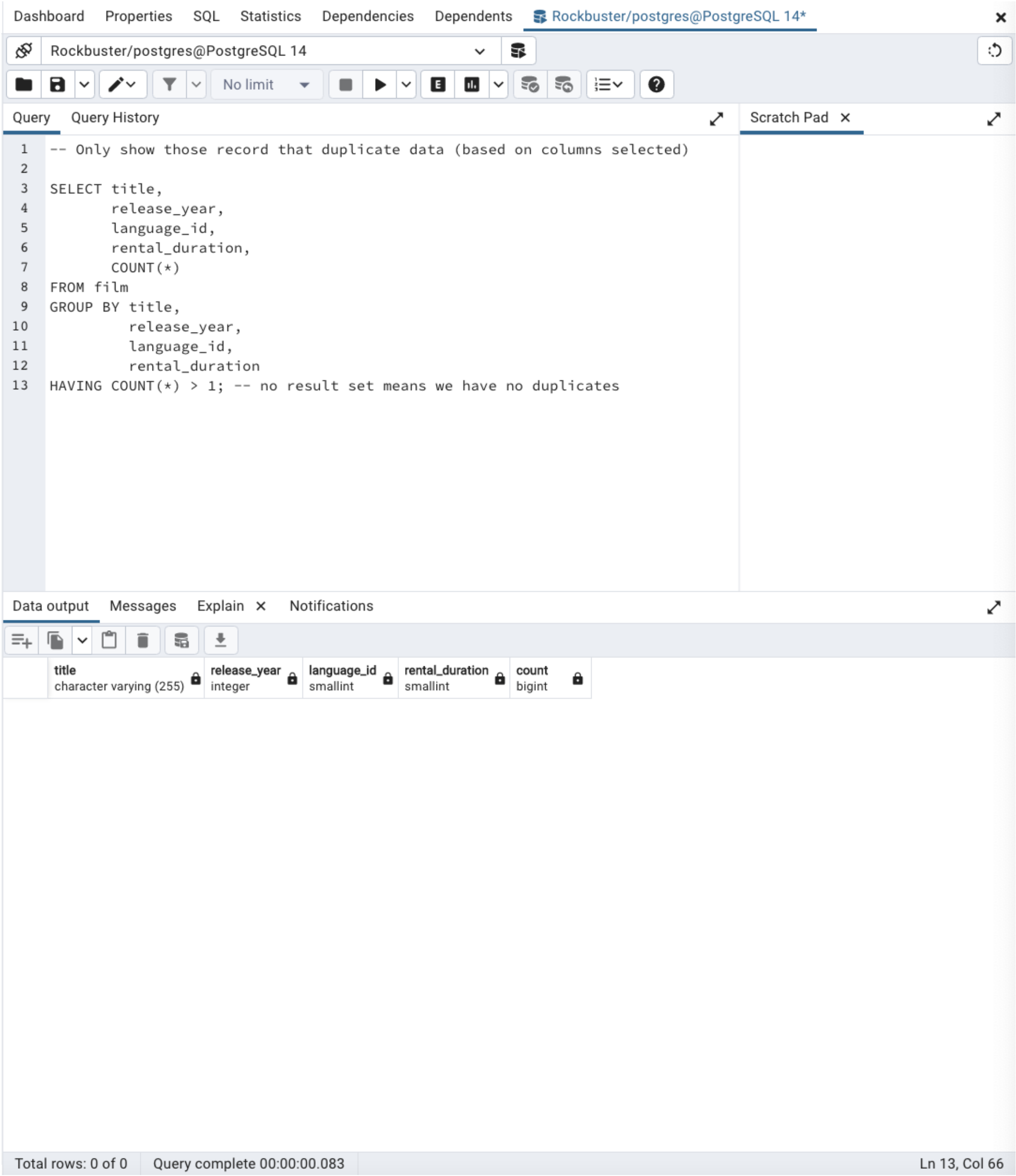
In this task you will calculate some descriptive statistics using the MIN, MAX, AVG, COUNT, SUM and MODE() aggregates discussed in this Exercise, and you will reflect on what you learned about data profiling back in Exercise 1.5: Data Profiling & Integrity.

Directions:

Rockbuster's database engineers have loaded some new data into the database, and your manager has asked you to clean and profile it. Follow the instructions below to complete their request:

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 "Answer 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)

Find the duplicate data (film):



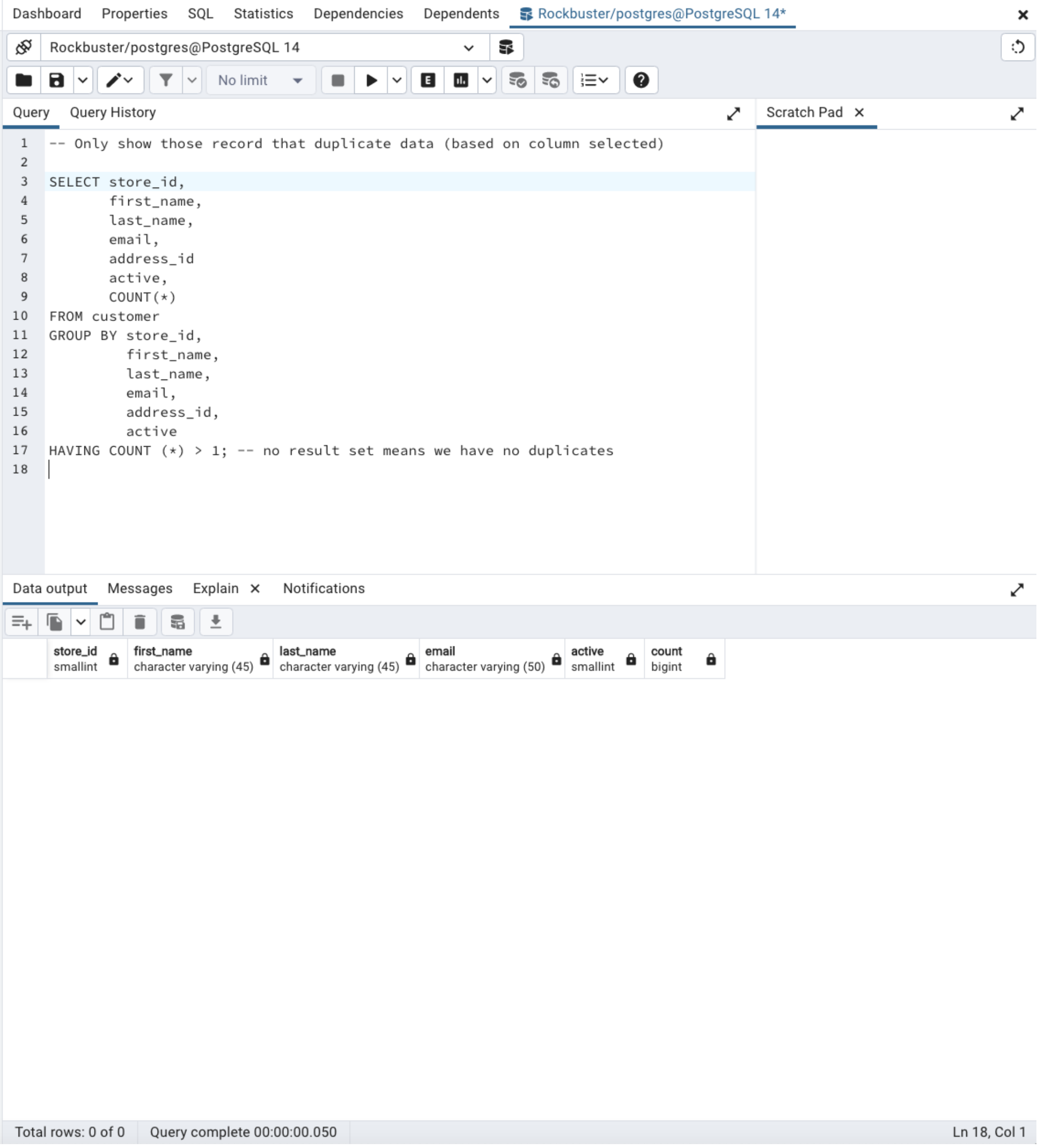
```
-- Only show those record that duplicate data (based on columns selected)

SELECT title,
       release_year,
       language_id,
       rental_duration,
       COUNT(*)
FROM film
GROUP BY title,
         release_year,
         language_id,
         rental_duration
HAVING COUNT(*) > 1; -- no result set means we have no duplicates
```

	title character varying (255)	release_year integer	language_id smallint	rental_duration smallint	count bigint

Total rows: 0 of 0   Query complete 00:00:00.083   Ln 13, Col 66

Find the duplicate data (customer):



```
-- Only show those record that duplicate data (based on column selected)

SELECT store_id,
       first_name,
       last_name,
       email,
       address_id
       active,
       COUNT(*)
FROM customer
GROUP BY store_id,
         first_name,
         last_name,
         email,
         address_id,
         active
HAVING COUNT (*) > 1; -- no result set means we have no duplicates
```

store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	active smallint	count bigint

Total rows: 0 of 0 Query complete 00:00:00.050 Ln 18, Col 1

There are no duplications on both film and customer data, as we can see from the Data Output above.

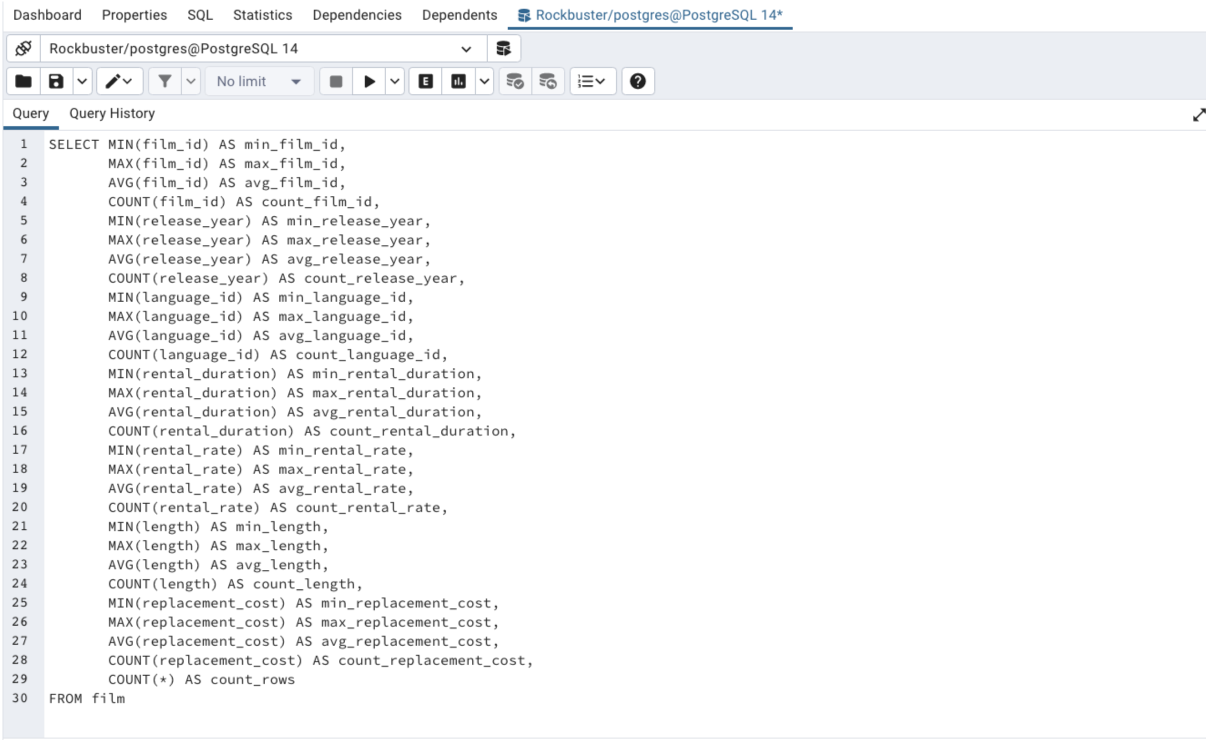
As described in Task 3.6, there are two ways that we can fix them if we find duplicate records in our database:

- Create a virtual table, known as a “view”, where you select only unique records – standard way
- Delete the duplicate record from the table or view

2. **Summarise 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-

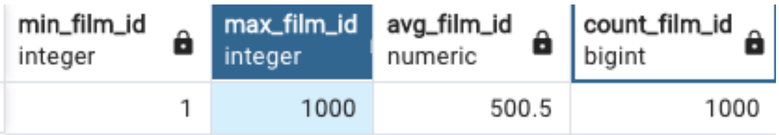
numerical columns, calculate the mode value. Copy-paste your SQL queries and their outputs into your answers document.

Film – numerical columns

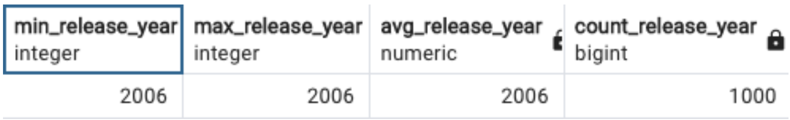


```sql
SELECT MIN(film_id) AS min_film_id,
       MAX(film_id) AS max_film_id,
       AVG(film_id) AS avg_film_id,
       COUNT(film_id) AS count_film_id,
       MIN(release_year) AS min_release_year,
       MAX(release_year) AS max_release_year,
       AVG(release_year) AS avg_release_year,
       COUNT(release_year) AS count_release_year,
       MIN(language_id) AS min_language_id,
       MAX(language_id) AS max_language_id,
       AVG(language_id) AS avg_language_id,
       COUNT(language_id) AS count_language_id,
       MIN(rental_duration) AS min_rental_duration,
       MAX(rental_duration) AS max_rental_duration,
       AVG(rental_duration) AS avg_rental_duration,
       COUNT(rental_duration) AS count_rental_duration,
       MIN(rental_rate) AS min_rental_rate,
       MAX(rental_rate) AS max_rental_rate,
       AVG(rental_rate) AS avg_rental_rate,
       COUNT(rental_rate) AS count_rental_rate,
       MIN(length) AS min_length,
       MAX(length) AS max_length,
       AVG(length) AS avg_length,
       COUNT(length) AS count_length,
       MIN(replacement_cost) AS min_replacement_cost,
       MAX(replacement_cost) AS max_replacement_cost,
       AVG(replacement_cost) AS avg_replacement_cost,
       COUNT(replacement_cost) AS count_replacement_cost,
       COUNT(*) AS count_rows
FROM film
```

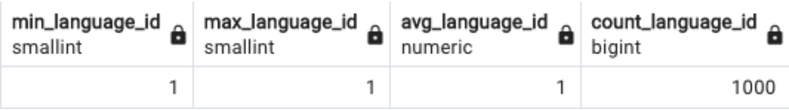
Film – numerical columns – Output (Note: The output has been split into a few screens due to the length of information)



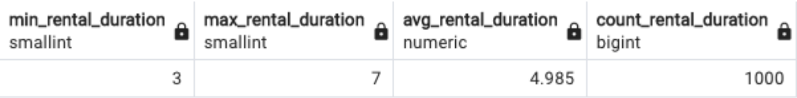
min_film_id integer	max_film_id integer	avg_film_id numeric	count_film_id bigint
1	1000	500.5	1000



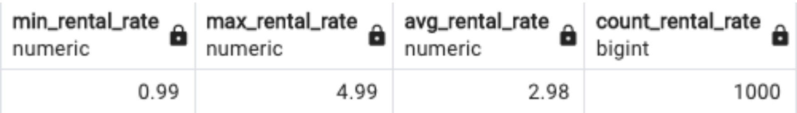
min_release_year integer	max_release_year integer	avg_release_year numeric	count_release_year bigint
2006	2006	2006	1000



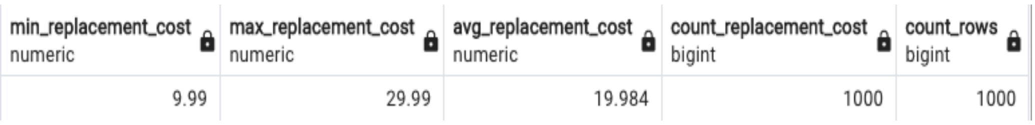
min_language_id smallint	max_language_id smallint	avg_language_id numeric	count_language_id bigint
1	1	1	1000



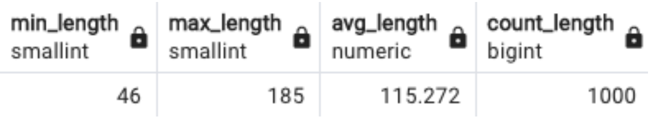
min_rental_duration smallint	max_rental_duration smallint	avg_rental_duration numeric	count_rental_duration bigint
3	7	4.985	1000



min_rental_rate numeric	max_rental_rate numeric	avg_rental_rate numeric	count_rental_rate bigint
0.99	4.99	2.98	1000

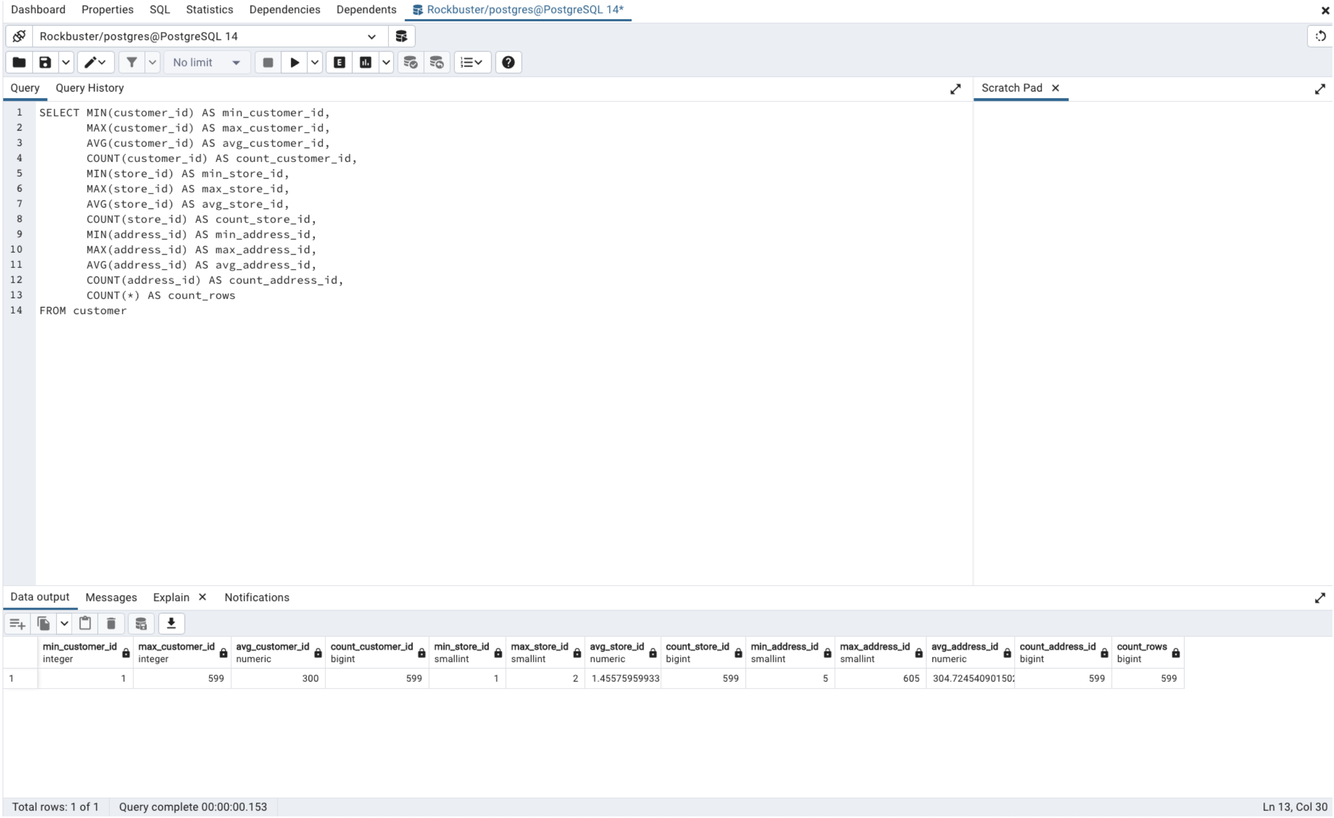


min_replacement_cost numeric	max_replacement_cost numeric	avg_replacement_cost numeric	count_replacement_cost bigint	count_rows bigint
9.99	29.99	19.984	1000	1000



min_length smallint	max_length smallint	avg_length numeric	count_length bigint
46	185	115.272	1000

Customer – numerical columns

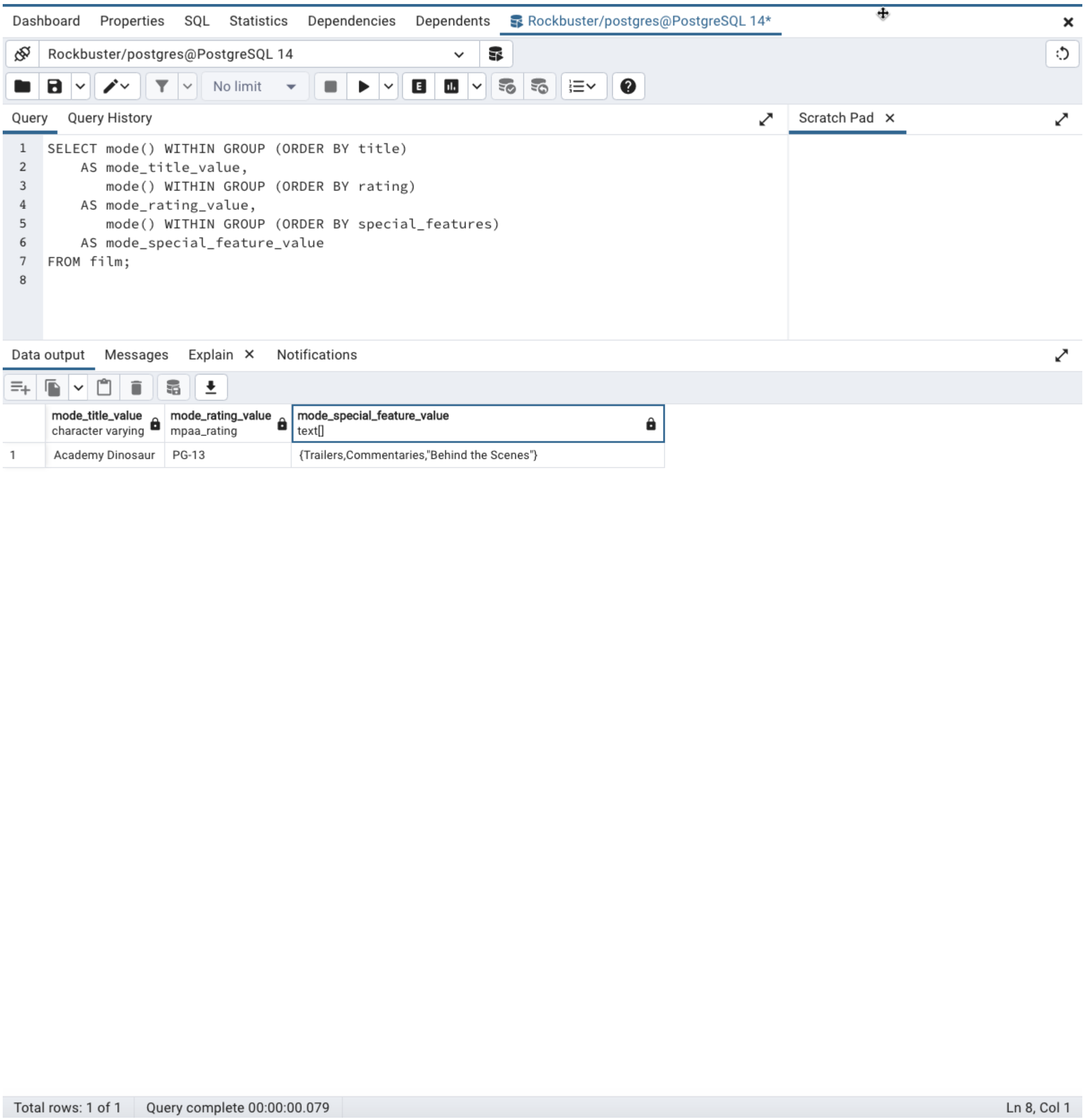


```
SELECT MIN(customer_id) AS min_customer_id,
       MAX(customer_id) AS max_customer_id,
       AVG(customer_id) AS avg_customer_id,
       COUNT(customer_id) AS count_customer_id,
       MIN(store_id) AS min_store_id,
       MAX(store_id) AS max_store_id,
       AVG(store_id) AS avg_store_id,
       COUNT(store_id) AS count_store_id,
       MIN(address_id) AS min_address_id,
       MAX(address_id) AS max_address_id,
       AVG(address_id) AS avg_address_id,
       COUNT(address_id) AS count_address_id,
       COUNT(*) AS count_rows
FROM customer
```

	min_customer_id integer	max_customer_id integer	avg_customer_id numeric	count_customer_id bigint	min_store_id smallint	max_store_id smallint	avg_store_id numeric	count_store_id bigint	min_address_id smallint	max_address_id smallint	avg_address_id numeric	count_address_id bigint	count_rows bigint
1	1	599	300	599	1	2	1.45575959933	599	5	605	304.72454090150	599	599

Total rows: 1 of 1 Query complete 00:00:00.153 Ln 13, Col 30

Film – non-numerical columns

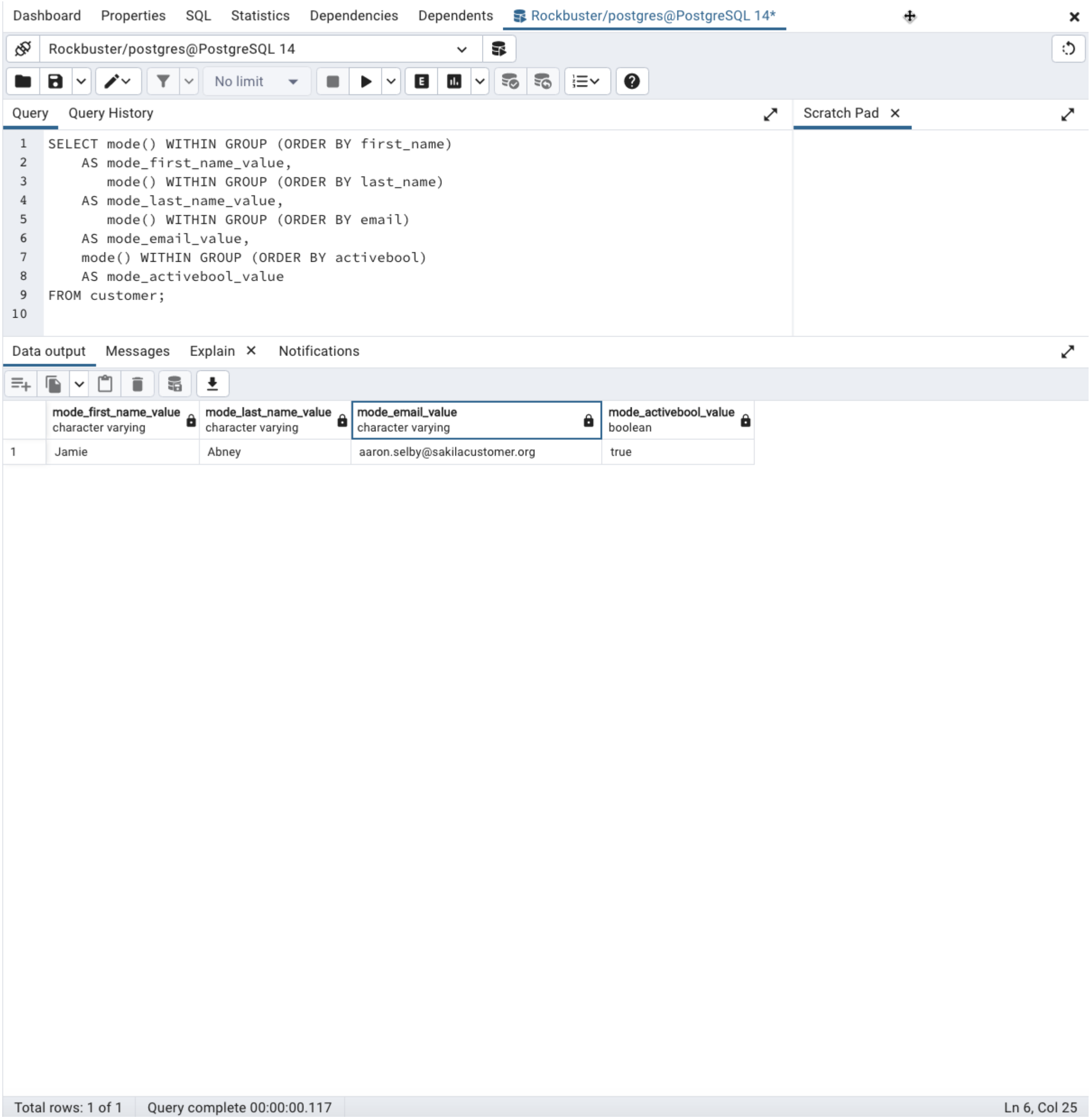


```
SELECT mode() WITHIN GROUP (ORDER BY title)
    AS mode_title_value,
       mode() WITHIN GROUP (ORDER BY rating)
    AS mode_rating_value,
       mode() WITHIN GROUP (ORDER BY special_features)
    AS mode_special_feature_value
FROM film;
```

	mode_title_value character varying	mode_rating_value mpaa_rating	mode_special_feature_value text[]
1	Academy Dinosaur	PG-13	{Trailers,Commentaries,"Behind the Scenes"}

Total rows: 1 of 1 Query complete 00:00:00.079 Ln 8, Col 1

Customer – non-numerical columns



```sql
SELECT mode() WITHIN GROUP (ORDER BY first_name)
    AS mode_first_name_value,
       mode() WITHIN GROUP (ORDER BY last_name)
    AS mode_last_name_value,
       mode() WITHIN GROUP (ORDER BY email)
    AS mode_email_value,
    mode() WITHIN GROUP (ORDER BY activebool)
    AS mode_activebool_value
FROM customer;
```

	mode_first_name_value character varying	mode_last_name_value character varying	mode_email_value character varying	mode_activebool_value boolean
1	Jamie	Abney	aaron.selby@sakilacustomer.org	true

Total rows: 1 of 1 Query complete 00:00:00.117 Ln 6, Col 25

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.

   Back in Achievement 1, it took me approximate 2 weeks to find the duplication and missing data with Excel. In this achievement, it took me less than 3 hours to find the repetition and missing data using the SQL function. Excel took me a lengthy time to average duplicated data and eliminate the duplication of data, while SQL could average the data by using the function. Clearly, the SQL is faster, easier and safer than Excel.

4. Save your “Answer 3.6” document as a PDF and upload it here for your tutor to review.