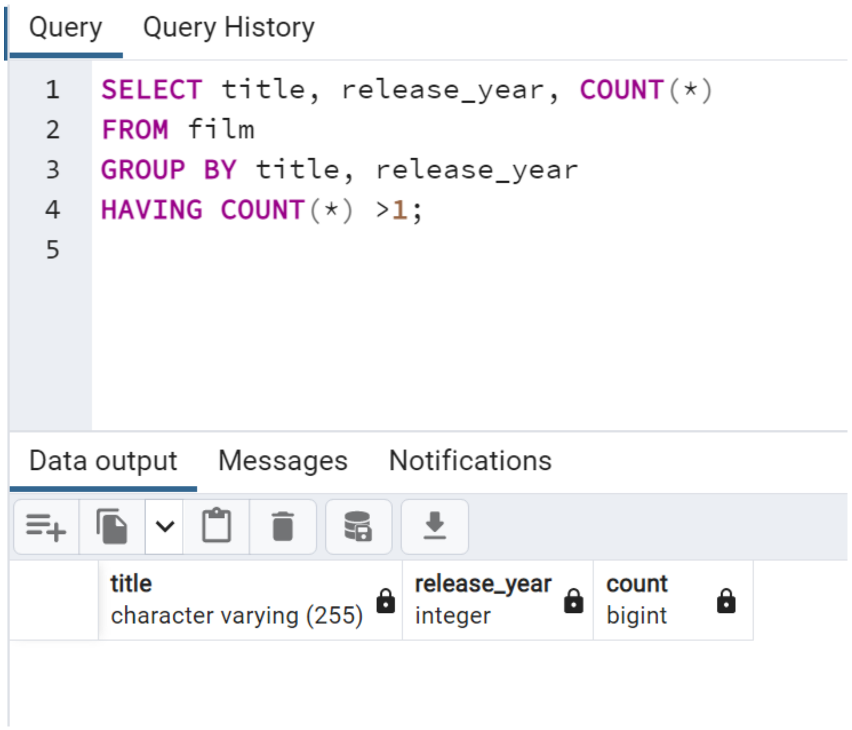
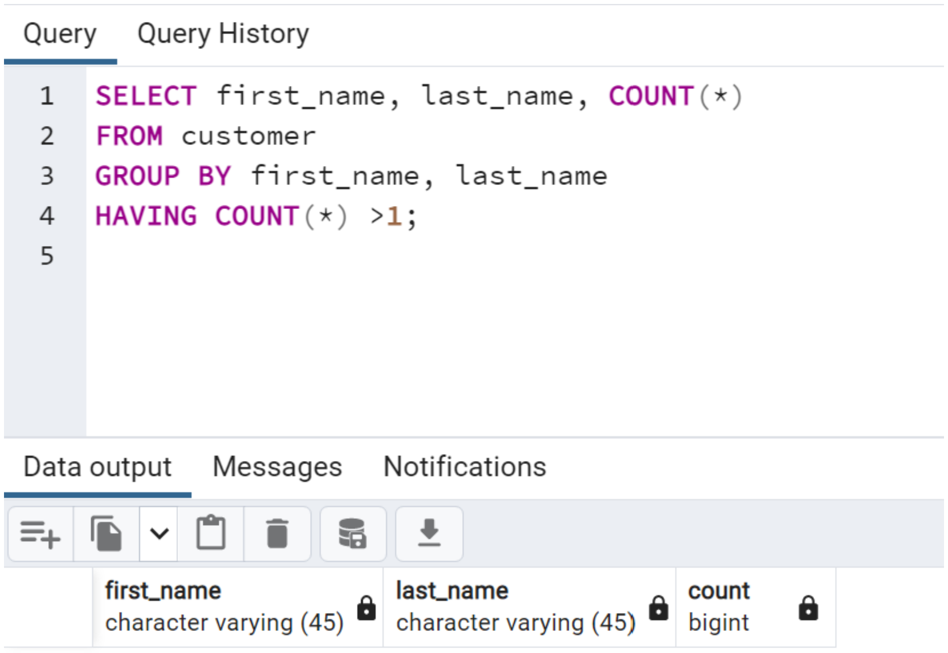
Find out if the film table and the customer table contain any dirty data, specifically non-uniform or duplicate data, or missing values

To find duplicates:



```sql
SELECT title, release_year, COUNT(*)
FROM film
GROUP BY title, release_year
HAVING COUNT(*) >1;
```

title character varying (255)	release_year integer	count bigint

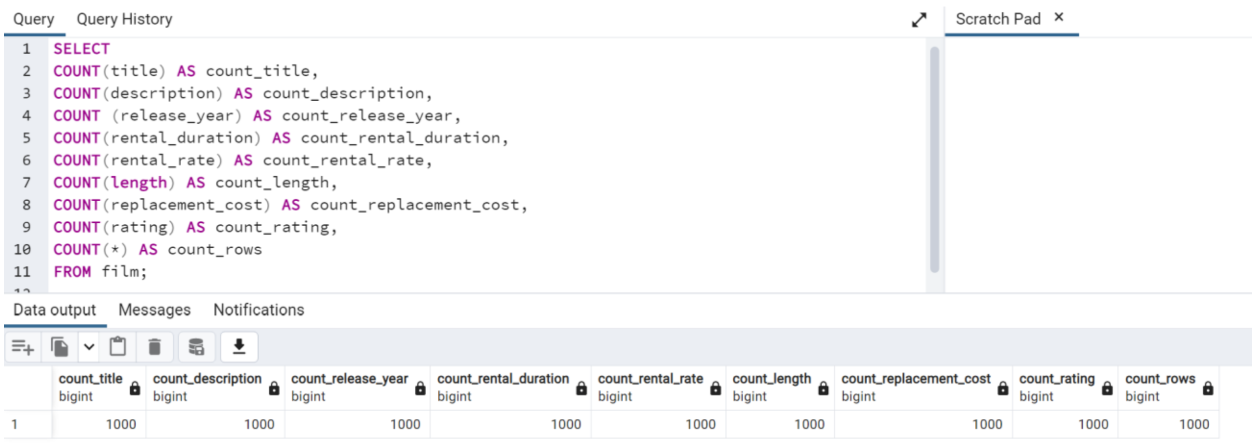


```sql
SELECT first_name, last_name, COUNT(*)
FROM customer
GROUP BY first_name, last_name
HAVING COUNT(*) >1;
```

first_name character varying (45)	last_name character varying (45)	count bigint

No duplicates were found in either table, but if there were, I could either delete the duplicates or create a view with unique records.

To find missing values:

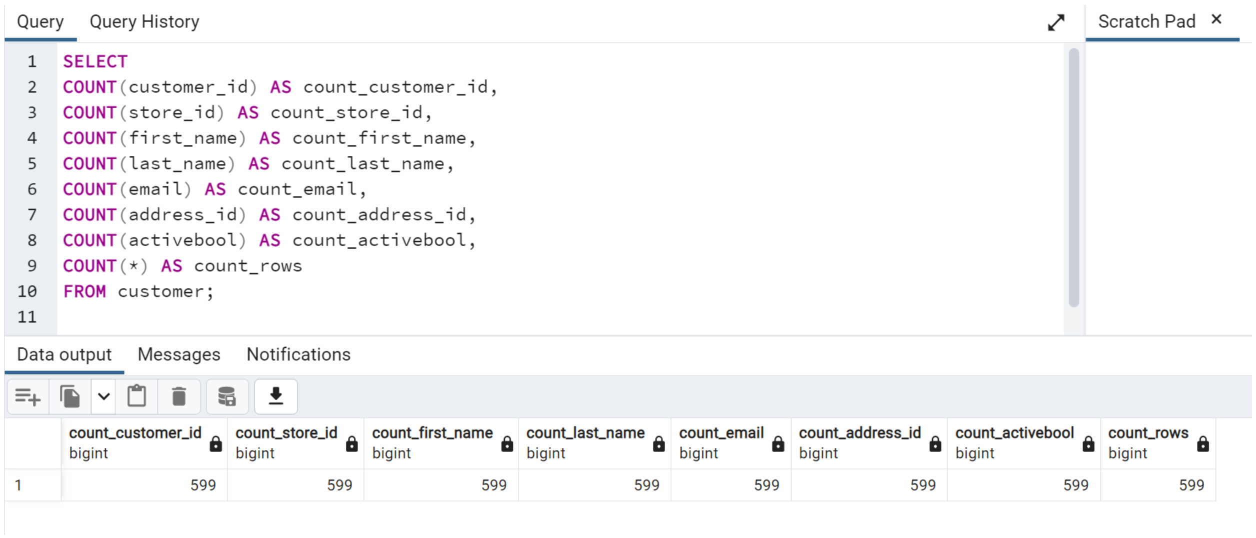


Query Query History Scratch Pad

```
SELECT
COUNT(title) AS count_title,
COUNT(description) AS count_description,
COUNT (release_year) AS count_release_year,
COUNT(rental_duration) AS count_rental_duration,
COUNT(rental_rate) AS count_rental_rate,
COUNT(length) AS count_length,
COUNT(replacement_cost) AS count_replacement_cost,
COUNT(rating) AS count_rating,
COUNT(*) AS count_rows
FROM film;
```

Data output Messages Notifications

	count_title bigint	count_description bigint	count_release_year bigint	count_rental_duration bigint	count_rental_rate bigint	count_length bigint	count_replacement_cost bigint	count_rating bigint	count_rows bigint
1	1000	1000	1000	1000	1000	1000	1000	1000	1000



Query Query History Scratch Pad

```
SELECT
COUNT(customer_id) AS count_customer_id,
COUNT(store_id) AS count_store_id,
COUNT(first_name) AS count_first_name,
COUNT(last_name) AS count_last_name,
COUNT(email) AS count_email,
COUNT(address_id) AS count_address_id,
COUNT(activebool) AS count_activebool,
COUNT(*) AS count_rows
FROM customer;
```

Data output Messages Notifications

	count_customer_id bigint	count_store_id bigint	count_first_name bigint	count_last_name bigint	count_email bigint	count_address_id bigint	count_activebool bigint	count_rows bigint
1	599	599	599	599	599	599	599	599

No missing values were found in either table. If there were a few missing values, I could fill them in with an average, and if there are a lot of missing values, I could ignore the column.

Use SQL to calculate descriptive statistics for both the film table and the customer table.

<u>Film table:</u>

SELECT MIN(rental_rate) AS min_rental_rate,

MAX(rental_rate) AS max_rental_rate,

AVG(rental_rate) AS avg_rental_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_id,

MAX(film_id) AS max_film_id,

AVG(film_id) AS avg_film_id,

MIN(language_id) AS min_language_id,

MAX(language_id) AS max_language_id,

AVG(language_id) AS avg_language_id,

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 year_value

FROM film
```

"min_rental_rate"	"max_rental_rate"	"avg_rental_rate"	"min_rental_duration"	"max_rental_duration"
0.99	4.99	2.9800000000000000	3	7

"avg_rental_duration"	"min_film_id"	"max_film_id"	"avg_film_id"	"min_language_id"
4.9850000000000000	1	1000	500.5000000000000000	1

"max_language_id"	"avg_language_id"	"min_length"	"max_length"	"avg_length"
1	1.00000000000000000000	46	185	115.2720000000000000

"min_replacement_cost"	"max_replacement_cost"	"avg_replacement_cost"	"rating_value"	"feature_value"	"year_value"
9.99	29.99	19.9840000000000000	"PG-13"	"{Trailers,Commentaries,""Behind the Scenes""}"	2006

<u>Customer Table:</u>

```
SELECT MIN(active) AS min_active,
MAX(active) AS max_active,
AVG(active) AS avg_active,
MIN(address_id) AS min_address_id,
MAX(address_id) AS max_address_id,
AVG(address_id) AS avg_address_id,
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,
mode() WITHIN GROUP (ORDER BY first_name) AS first_name_value,
mode() WITHIN GROUP (ORDER BY last_name) AS last_name_value,
mode() WITHIN GROUP (ORDER BY email) AS email_value
FROM customer;
```

"min_active"	"max_active"	"avg_active"	"min_address_id"	"max_address_id"
0	1	0.97495826377295492487	5	605

"avg_address_id"	"min_customer_id"	"max_customer_id"	"avg_customer_id"	"min_store_id"	"max_store_id"	"avg_store_id"	"first_name_value"	"last_name_value"	"email_value"
304.7245409015025042	1	599	300.0000000000000000	1	2	1.4557595993322204	"Jamie"	"Abney"	"aaron.selby@sakilacustomer.org"