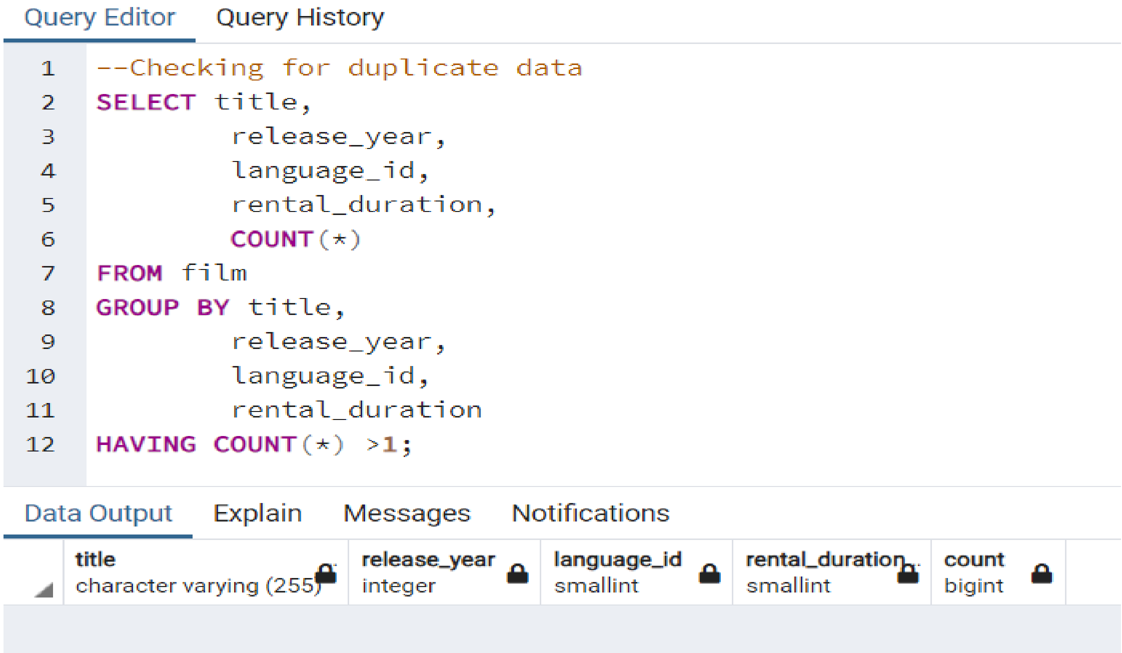
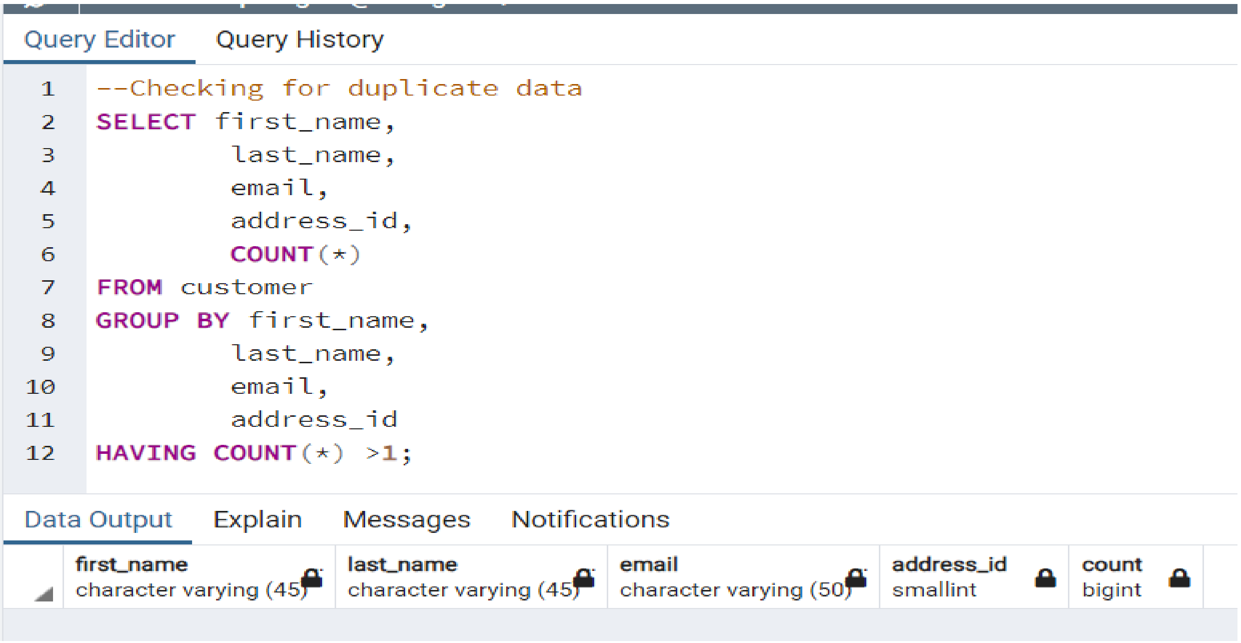
3.6: Summarizing & Cleaning Data in SQL

1. Check for and clean dirty data:



```sql
--Checking for duplicate data
SELECT title,
        release_year,
        language_id,
        rental_duration,
        COUNT(*)
FROM film
GROUP BY title,
        release_year,
        language_id,
        rental_duration
HAVING COUNT(*) >1;
```

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



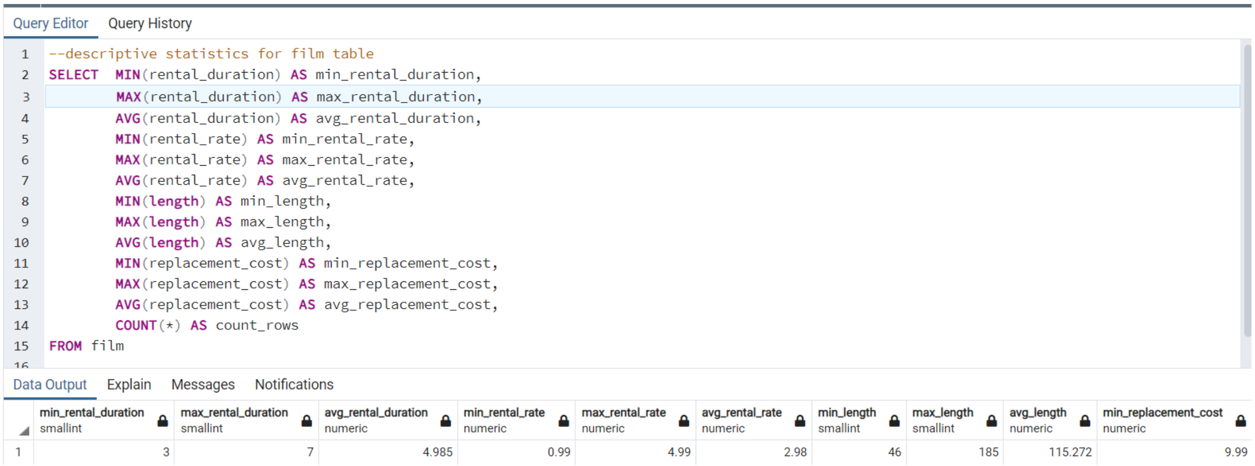
```sql
--Checking for duplicate data
SELECT first_name,
        last_name,
        email,
        address_id,
        COUNT(*)
FROM customer
GROUP BY first_name,
        last_name,
        email,
        address_id
HAVING COUNT(*) >1;
```

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

If the data needed to be cleaned, I could do so by creating a view table showing only unique records or I could delete the duplicate records. If I don’t have permissions to update, delete, or create views then I could write a query that only returns unique records by using group by or distinct.

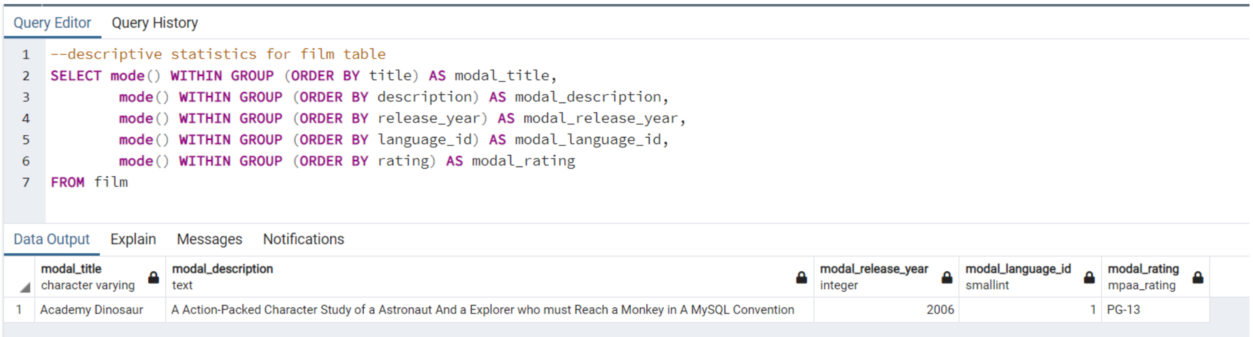
2. Summarize your data:

Film Table



```sql
--descriptive statistics for film table
SELECT  MIN(rental_duration) AS min_rental_duration,
        MAX(rental_duration) AS max_rental_duration,
        AVG(rental_duration) AS avg_rental_duration,
        MIN(rental_rate) AS min_rental_rate,
        MAX(rental_rate) AS max_rental_rate,
        AVG(rental_rate) AS avg_rental_rate,
        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,
        COUNT(*) AS count_rows
FROM film
```

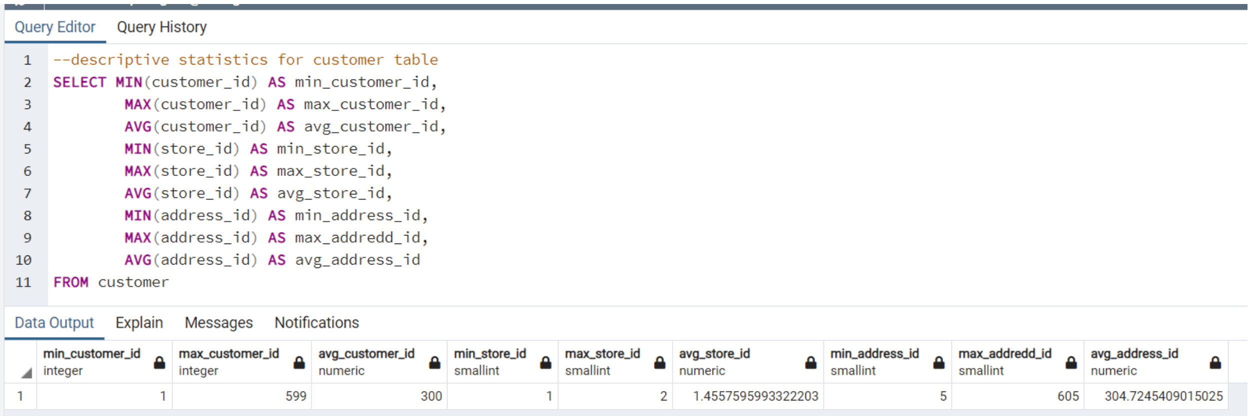
	min_rental_duration smallint	max_rental_duration smallint	avg_rental_duration numeric	min_rental_rate numeric	max_rental_rate numeric	avg_rental_rate numeric	min_length smallint	max_length smallint	avg_length numeric	min_replacement_cost numeric
1	3	7	4.985	0.99	4.99	2.98	46	185	115.272	9.99



```sql
--descriptive statistics for film table
SELECT mode() WITHIN GROUP (ORDER BY title) AS modal_title,
        mode() WITHIN GROUP (ORDER BY description) AS modal_description,
        mode() WITHIN GROUP (ORDER BY release_year) AS modal_release_year,
        mode() WITHIN GROUP (ORDER BY language_id) AS modal_language_id,
        mode() WITHIN GROUP (ORDER BY rating) AS modal_rating
FROM film
```

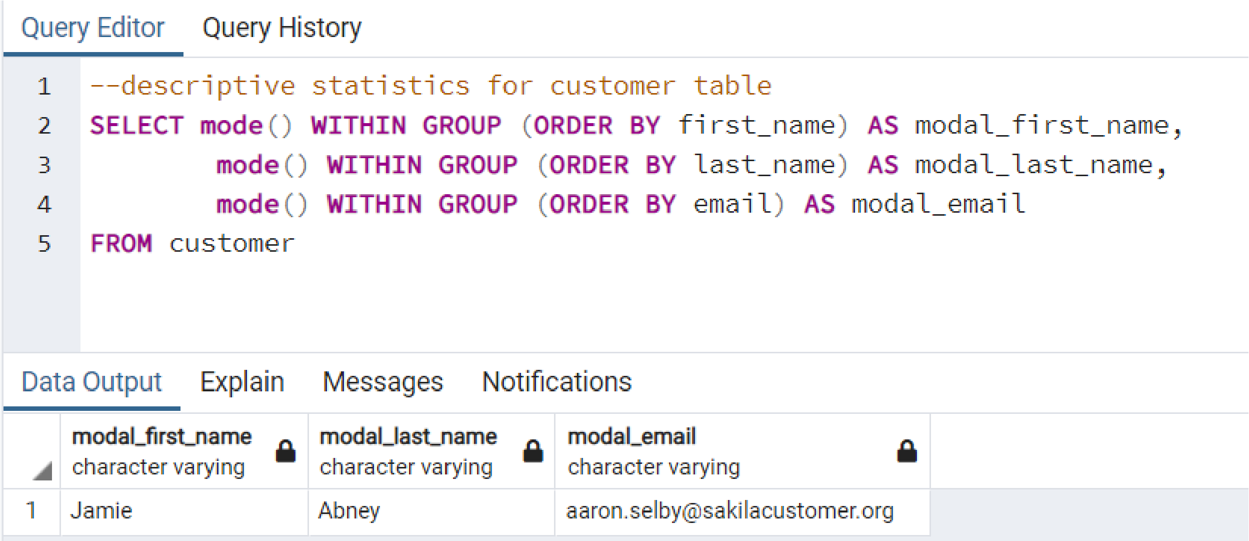
	modal_title character varying	modal_description text	modal_release_year integer	modal_language_id smallint	modal_rating mpaa_rating
1	Academy Dinosaur	A Action-Packed Character Study of a Astronaut And a Explorer who must Reach a Monkey in A MySQL Convention	2006	1	PG-13

Customer Table



```sql
--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_addredd_id,
        AVG(address_id) AS avg_address_id
FROM customer
```

	min_customer_id integer	max_customer_id integer	avg_customer_id numeric	min_store_id smallint	max_store_id smallint	avg_store_id numeric	min_address_id smallint	max_addredd_id smallint	avg_address_id numeric
1	1	599	300	1	2	1.4557595993322203	5	605	304.7245409015025



```
--descriptive statistics for customer table
SELECT mode() WITHIN GROUP (ORDER BY first_name) AS modal_first_name,
        mode() WITHIN GROUP (ORDER BY last_name) AS modal_last_name,
        mode() WITHIN GROUP (ORDER BY email) AS modal_email
FROM customer
```

	modal_first_name character varying	modal_last_name character varying	modal_email character varying
1	Jamie	Abney	aaron.selby@sakilacustomer.org

3. Reflect on your work:

When working with smaller data sets, I feel excel is easier for me as typing the query out correctly in SQL takes more time than getting the info in excel. When working with larger data sets the ability to write the query, add comments of what your doing, always having the query there in history if you need to go back to it, and how quickly it returns results makes SQL more effective in data profiling.