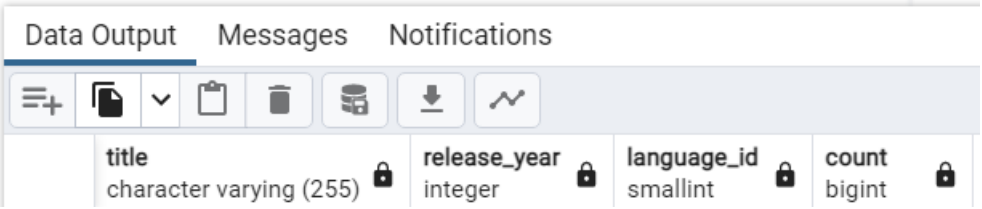
3.6: Summarizing & Cleaning Data in SQL

Duplicates

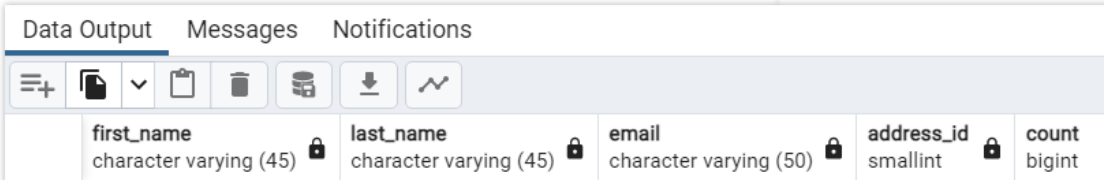
Film table

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



Customer table

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



In none of the tables could be duplicates identified. However, in case that yes, I could create a VIEW with only unique records. Another option would be to remove duplicate data with DELETE. If I don't have permission to do that, I`ll need to write a query that returns only unique records. There are 2 ways to do this – GROUP BY and DISTINCT.

Non-Uniform Data

Film Table

Query Query History

```
1  SELECT release_year, language_id, rating
2  FROM film
3  GROUP BY release_year, language_id, rating;
```

Data Output Messages Notifications

	release_year integer	language_id smallint	rating mpaa_rating
1	2006	1	PG-13
2	2006	1	NC-17
3	2006	1	G
4	2006	1	PG
5	2006	1	R

Customer table

```
1  SELECT DISTINCT activebool, create_date
2  FROM customer;
```

Data Output Messages Notifications

	activebool boolean	create_date date
	true	2006-02-14

No non-uniform could be identified. In case that yes, the UPDATE can be used to make values consistent.

Missing Values

Film Table

```
SELECT title, description, language_id, release_year
FROM film
WHERE release_year
IS null;
```

Data Output | Messages | Notifications

title	description	language_id	release_year
character varying (255)	text	smallint	integer

```
SELECT first_name, last_name, email
FROM customer
WHERE email
IS null;
```

Data Output | Messages | Notifications

first_name	last_name	email
character varying (45)	character varying (45)	character varying (50)

No missing values could be identified. If a column contains a high number of missing values, the column should be excluded from the analyse by omitting it from the SELECT command. If a column contains a low amount of missing values, these missing values can be imputed with an estimate, such as column average.

2. Descriptive Statistics

Film table

```
SELECT
        MIN(rental_duration) AS min_duration,
        MAX(rental_duration) AS max_duration,
        AVG(rental_duration) AS avg_duration,

        MIN(rental_rate) AS min_rent,
        MAX(rental_rate) AS max_rent,
        AVG(rental_rate) AS avg_rent,

        MIN(length) AS min_length,
        MAX(length) AS max_length,
        AVG(length) AS avg_length,

        MIN(replacement_cost) AS min_replacement,
        MAX(replacement_cost) AS max_replacement,
        AVG(replacement_cost) AS avg_replacment,

        MODE() WITHIN GROUP(ORDER BY release_year) AS mode_year,
        MODE() WITHIN GROUP(Order BY language_id) AS mode_language_id,
        MODE() WITHIN GROUP (Order BY rating) AS mode_rating

FROM film
```

Data Output | Messages | Notifications

min_duration smallint	max_duration smallint	avg_duration numeric	min_rent numeric	max_rent numeric	avg_rent numeric	min_length smallint	max_length smallint	avg_length numeric	min_replacement numeric	max_replacement numeric	avg_replacment numeric	mode_year integer	mode_language_id smallint	mode mpaa
3	7	4.985	0.99	4.99	2.98	46	185	115.272	9.99	29.99	19.984	2006	1	PG-13

Customer table

```
SELECT
        MIN(customer_id) AS min_cust_id,
        MAX(customer_id) AS customer_id,
        AVG(customer_id) AS customer_id,

        MIN(store_id) AS store_id,
        MAX(store_id) AS store_id,
        MODE() WITHIN GROUP (ORDER BY store_id) AS mode_store_id,

        MIN(address_id) AS address_id,
        MAX(address_id) AS address_id,

        MIN(create_date) AS create_date,
        MAX(create_date) AS create_date,
        MODE() WITHIN GROUP (ORDER BY create_date) AS mode_create_date,

        MIN(last_update) AS min_last_update,
        MAX(last_update) AS last_update,
        MODE() WITHIN GROUP (ORDER BY last_update) AS last_update,
        mode() WITHIN GROUP (ORDER BY activebool) AS mode_activebool


FROM customer;
```

Data Output | Messages | Notifications

min_cust_id integer	customer_id integer	customer_id numeric	store_id smallint	store_id smallint	mode_store_id smallint	address_id smallint	address_id smallint	create_date date	create_date date	mode_create_date date	min_last_update timestamp without time zone	last_update timestamp without time zone	las tim
1	599	300	1	2	1	5	605	2006-02-14	2006-02-14	2006-02-14	2013-05-26 14:49:45.738	2013-05-26 14:49:45.738	20

3. Reflection

SQL makes data profiling very simple and quick (once you get used to the Synthax and the logic). Especially when data set is large, SQL provides a very efficient way to work with a data set, especially when it's about descriptive statistics. You can quickly identify the completenss and uniqueness of the dataset with DISTICNT and GROUP BY. However, since I am used to work with Excel already for several years, SQL is more complicated and slower for me. Especially the work with a smaller dataset would be faster with Excel. It's literally like learning a new language – first you need to set a steady base (the Syntax) and then it's practice and more practice to become fluent.