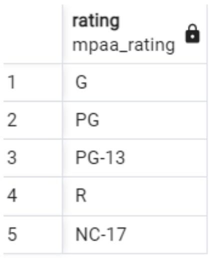
3.6 Summarizing & Cleaning Data in SQL

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).*

Looking for non-unique values:

```
--looking for non-uniform data
SELECT DISTINCT rating
FROM film
GROUP BY rating
```

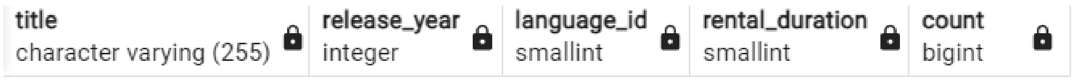


	rating mpaa_rating
1	G
2	PG
3	PG-13
4	R
5	NC-17

If there were some non-uniform data, one could check for inconsistencies using GROUP BY and DISTINCT commands and, if there any, UPDATE the values.

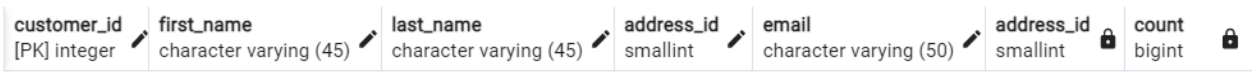
Looking for duplicate data:

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



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

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

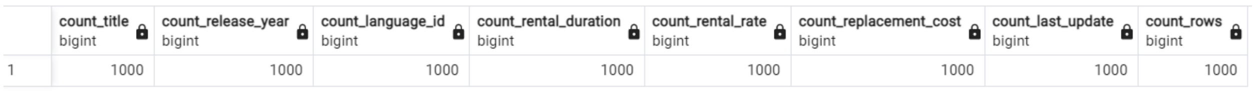


customer_id [PK] integer	first_name character varying (45)	last_name character varying (45)	address_id smallint	email character varying (50)	address_id smallint	count bigint

If there were any duplicates data, the problem could be solved in two ways:
- Creation of a virtual table, known as a "view," where only unique records are selected.
- Deletion of the duplicate record from the table or view.

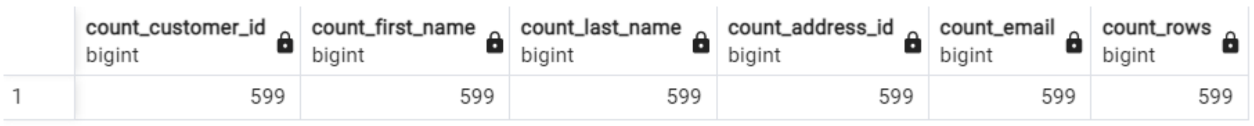
Looking for missing data:

```
SELECT COUNT (title) AS count_title,
       COUNT (release_year) AS count_release_year,
       COUNT (language_id) AS count_language_id,
       COUNT (rental_duration) AS count_rental_duration,
       COUNT (rental_rate) AS count_rental_rate,
       COUNT (replacement_cost) AS count_replacement_cost,
       COUNT (last_update) AS count_last_update,
       COUNT (*) AS count_rows
FROM film
```



	count_title bigint	count_release_year bigint	count_language_id bigint	count_rental_duration bigint	count_rental_rate bigint	count_replacement_cost bigint	count_last_update bigint	count_rows bigint
1	1000	1000	1000	1000	1000	1000	1000	1000

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



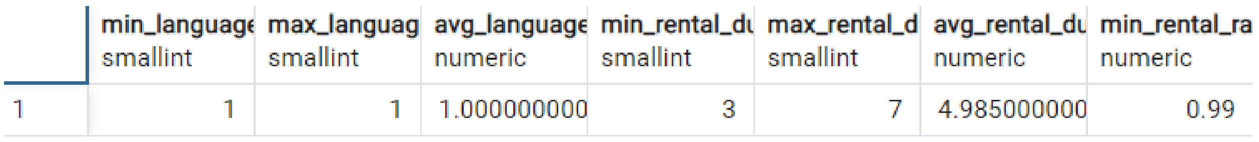
	count_customer_id bigint	count_first_name bigint	count_last_name bigint	count_address_id bigint	count_email bigint	count_rows bigint
1	599	599	599	599	599	599

If there were missing data, the ways to solve the problem could be:
- Ignore columns with a high percentage of missing values.
- Impute the missing values using statistical methods.

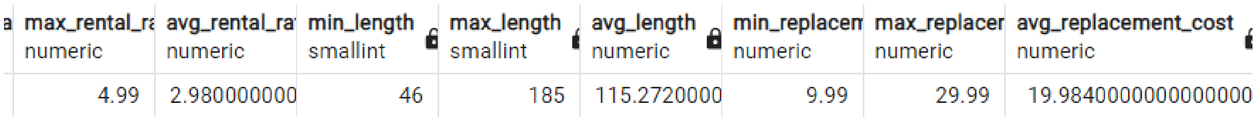
2. ***Summarize your data:*** *Use SQL to calculate descriptive statistics for both the film table and the customer table. For numerical columns, this means finding the minimum, maximum, and average values. For non-numerical columns, calculate the mode value. Copy-paste your SQL queries and their outputs into your answers document.*

Film table:

```
--descriptive statistics for numerical columns for film
table

SELECT MIN (language_id) AS min_language_id,
       MAX (language_id) AS max_language_id,
       AVG (language_id) AS avg_language_id,
       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
FROM film
```

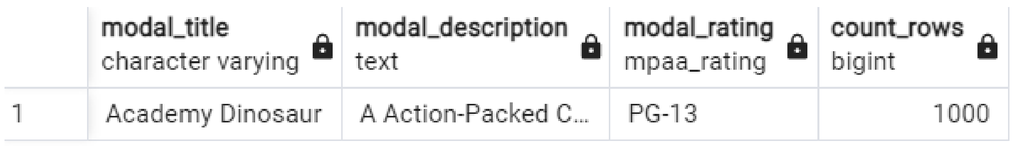


	min_language	max_languag	avg_language	min_rental_du	max_rental_d	avg_rental_du	min_rental_ra
	smallint	smallint	numeric	smallint	smallint	numeric	numeric
1	1	1	1.000000000	3	7	4.985000000	0.99



max_rental_ra	avg_rental_ra	min_length	max_length	avg_length	min_replacem	max_replacer	avg_replacement_cost
numeric	numeric	smallint	smallint	numeric	numeric	numeric	numeric
4.99	2.980000000	46	185	115.2720000	9.99	29.99	19.9840000000000000

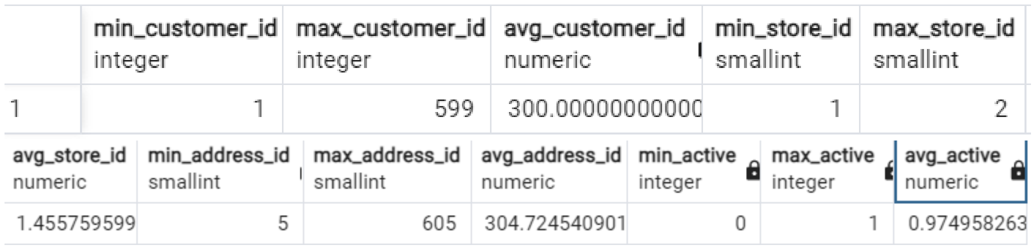
```
--descriptive statistics for non-numerical columns for film
table
SELECT mode () WITHIN GROUP (ORDER BY title) AS
modal_title,
       mode () WITHIN GROUP (ORDER BY description) AS
modal_description,
       mode () WITHIN GROUP (ORDER BY rating) AS
modal_rating,
       COUNT (*) AS count_rows
FROM film
```



	modal_title character varying	modal_description text	modal_rating mpaa_rating	count_rows bigint
1	Academy Dinosaur	A Action-Packed C...	PG-13	1000

Customer table:

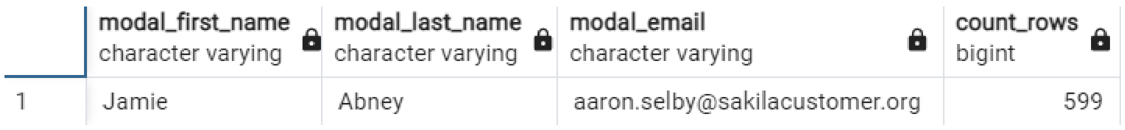
```
--descriptive statistics for numerical columns for customer
table

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 (active) AS min_active,
       MAX (active) AS max_active,
       AVG (active) AS avg_active
FROM customer
```



	min_customer_id integer	max_customer_id integer	avg_customer_id numeric	min_store_id smallint	max_store_id smallint
1	1	599	300.00000000000	1	2

avg_store_id numeric	min_address_id smallint	max_address_id smallint	avg_address_id numeric	min_active integer	max_active integer	avg_active numeric
1.455759599	5	605	304.724540901	0	1	0.974958263

```
--descriptive statistics for non-numerical columns 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,
       COUNT (*) AS count_rows
FROM customer
```



	modal_first_name character varying	modal_last_name character varying	modal_email character varying	count_rows bigint
1	Jamie	Abney	aaron.selby@sakilacustomer.org	599

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.*

   For data profiling SQL works better than Excel, since it allows a faster manipulation with large volume of data. Knowing SQL syntax one can save time when retrieving information using SQL. Besides, SQL has more instruments for data manipulations and more ways for data safety and security (like, DSP model, or using VIEWs).