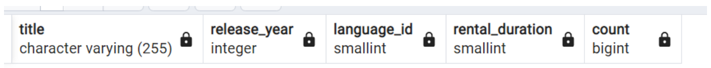
3.6 Summarizing and Cleaning Data in SQL

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).

Film table

Duplicate values

```
SELECT title,
       release_year,
       language_id,
       rental_duration,
       COUNT(*)
FROM film
GROUP BY title,
         release_year,
         language_id,
         rental_duration
HAVING COUNT(*) >1; --no
```

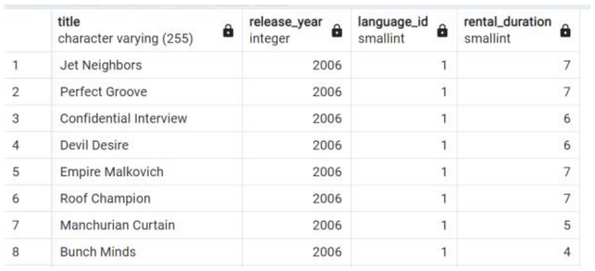


title character varying (255)	release_year integer	language_id smallint	rental_duration smallint	count bigint

- No duplicate values since there isn’t any count bigger than one.

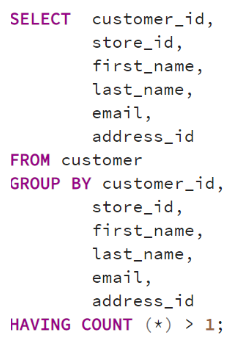
Non uniform data

```
SELECT DISTINCT title,
                release_year,
                language_id,
                rental_duration
FROM film;
```

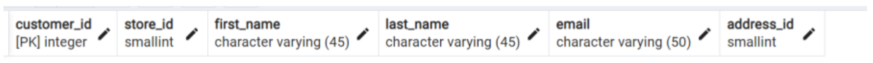


	title character varying (255)	release_year integer	language_id smallint	rental_duration smallint
1	Jet Neighbors	2006	1	7
2	Perfect Groove	2006	1	7
3	Confidential Interview	2006	1	6
4	Devil Desire	2006	1	6
5	Empire Malkovich	2006	1	7
6	Roof Champion	2006	1	7
7	Manchurian Curtain	2006	1	5
8	Bunch Minds	2006	1	4

Customer table



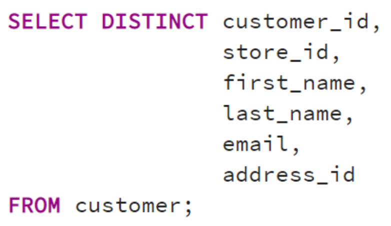
```
SELECT  customer_id,
        store_id,
        first_name,
        last_name,
        email,
        address_id
FROM customer
GROUP BY customer_id,
        store_id,
        first_name,
        last_name,
        email,
        address_id
HAVING COUNT (*) > 1;
```



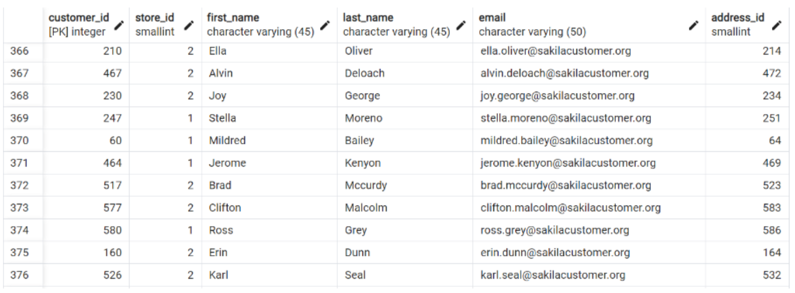
customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint

- No duplicate values in the customer table

Non uniform data



```
SELECT DISTINCT customer_id,
                store_id,
                first_name,
                last_name,
                email,
                address_id
FROM customer;
```



	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint
366	210	2	Ella	Oliver	ella.oliver@sakilacustomer.org	214
367	467	2	Alvin	Deloach	alvin.deloach@sakilacustomer.org	472
368	230	2	Joy	George	joy.george@sakilacustomer.org	234
369	247	1	Stella	Moreno	stella.moreno@sakilacustomer.org	251
370	60	1	Mildred	Bailey	mildred.bailey@sakilacustomer.org	64
371	464	1	Jerome	Kenyon	jerome.kenyon@sakilacustomer.org	469
372	517	2	Brad	Mccurdy	brad.mccurdy@sakilacustomer.org	523
373	577	2	Clifton	Malcolm	clifton.malcolm@sakilacustomer.org	583
374	580	1	Ross	Grey	ross.grey@sakilacustomer.org	586
375	160	2	Erin	Dunn	erin.dunn@sakilacustomer.org	164
376	526	2	Karl	Seal	karl.seal@sakilacustomer.org	532

In general, there aren't any duplicated values neither in the customer table nor the film table. Also, the Distinct function was to run to analyze if there is any strange value.

In general, all the values present a similar structure and are similar in the data type expected in each category.

In case we needed to clean data, after checking if there are any duplicate values, for deleting data we can do the following:

- Create a view which is a virtual table that includes only the records needed for the analysis.

- Select unique values by using GROUP BY:
  ```
  SELECT customer_id,
          store_id,
          first_name
  FROM customer
  GROUP BY   customer_id,
             store_id,
             first_name;
  ```

- Select unique values using DISTINCT:
  ```
  SELECT DISTINCT customer_id,
  ```

```
                    store_id,
                    first_name
FROM customer;
```

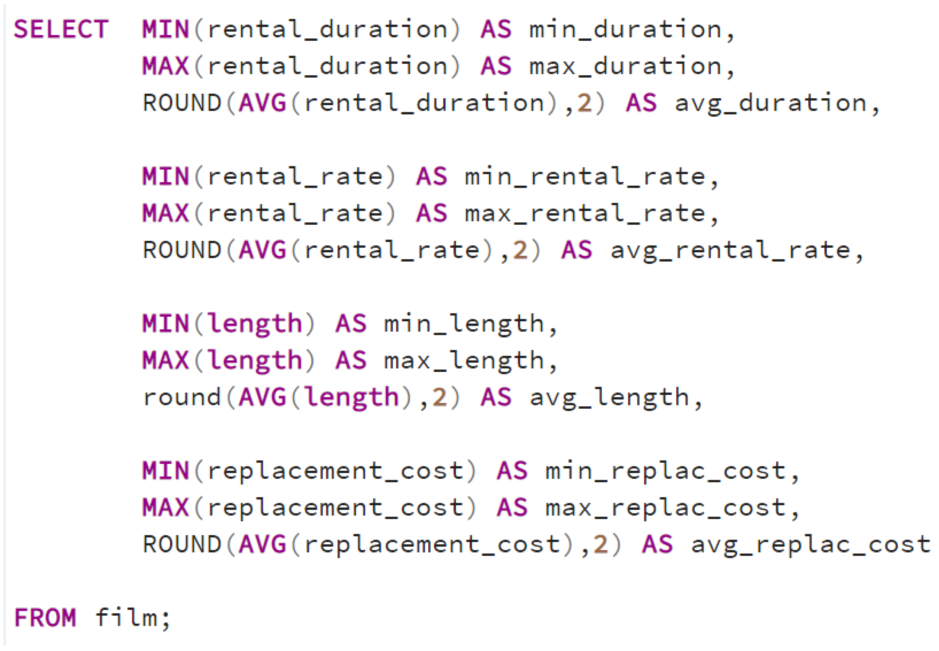
- Update the value by using UPDATE

```
UPDATE customer
SET address_id = '2'
WHERE address_id IN ('11 California Av')
```

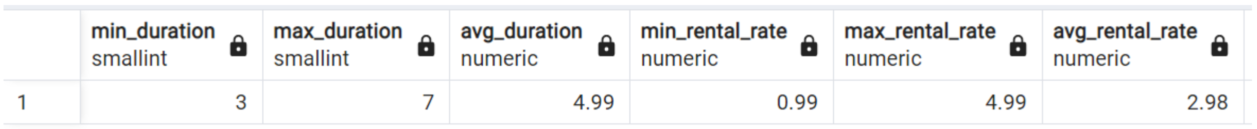
2. **Summarize your data:** Use SQL to calculate descriptive statistics for both the film table and the customer table. For numerical columns, this means finding the minimum, maximum, and average values. For non-numerical columns, calculate the mode value. Copy-paste your SQL queries and their outputs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

Film table

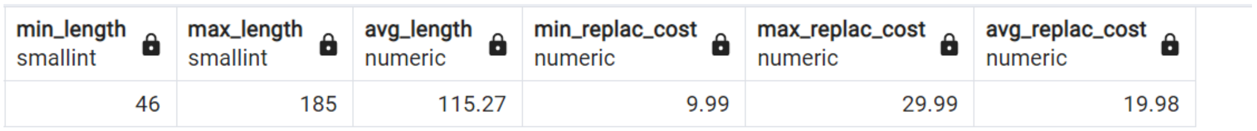
Numerical columns: rental_duration, rental_rate, length, replacement_cost



```
SELECT  MIN(rental_duration) AS min_duration,
        MAX(rental_duration) AS max_duration,
        ROUND(AVG(rental_duration),2) AS avg_duration,

        MIN(rental_rate) AS min_rental_rate,
        MAX(rental_rate) AS max_rental_rate,
        ROUND(AVG(rental_rate),2) AS avg_rental_rate,

        MIN(length) AS min_length,
        MAX(length) AS max_length,
        round(AVG(length),2) AS avg_length,

        MIN(replacement_cost) AS min_replac_cost,
        MAX(replacement_cost) AS max_replac_cost,
        ROUND(AVG(replacement_cost),2) AS avg_replac_cost

FROM film;
```

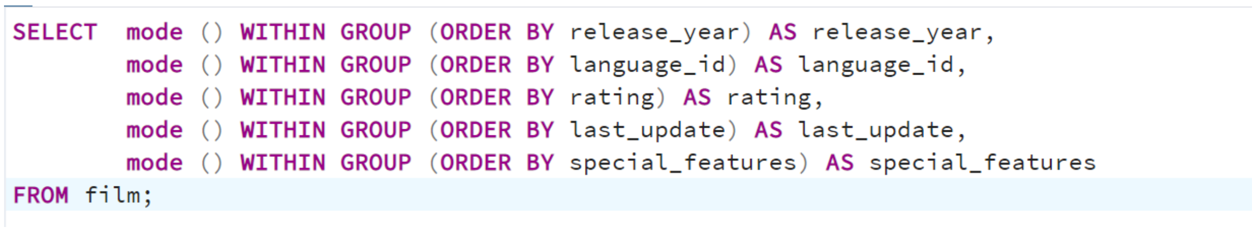


	min_duration smallint	max_duration smallint	avg_duration numeric	min_rental_rate numeric	max_rental_rate numeric	avg_rental_rate numeric
1	3	7	4.99	0.99	4.99	2.98

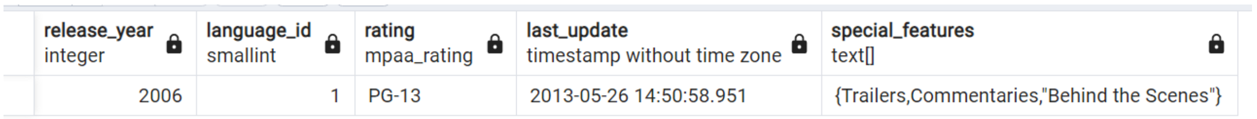


min_length smallint	max_length smallint	avg_length numeric	min_replac_cost numeric	max_replac_cost numeric	avg_replac_cost numeric
46	185	115.27	9.99	29.99	19.98

<u>Non-numerical</u>: release_year, rating,



```
SELECT  mode () WITHIN GROUP (ORDER BY release_year) AS release_year,
        mode () WITHIN GROUP (ORDER BY language_id) AS language_id,
        mode () WITHIN GROUP (ORDER BY rating) AS rating,
        mode () WITHIN GROUP (ORDER BY last_update) AS last_update,
        mode () WITHIN GROUP (ORDER BY special_features) AS special_features
FROM film;
```

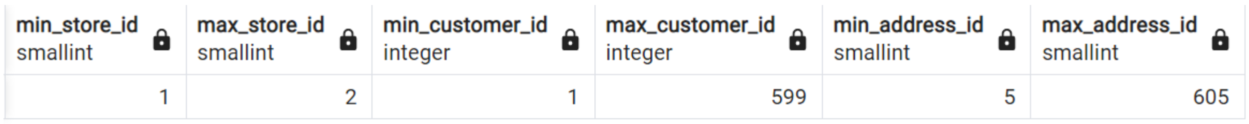


	release_year integer	language_id smallint	rating mpaa_rating	last_update timestamp without time zone	special_features text[]
	2006	1	PG-13	2013-05-26 14:50:58.951	{Trailers,Commentaries,"Behind the Scenes"}

Customer table

<u>Numerical columns:</u>

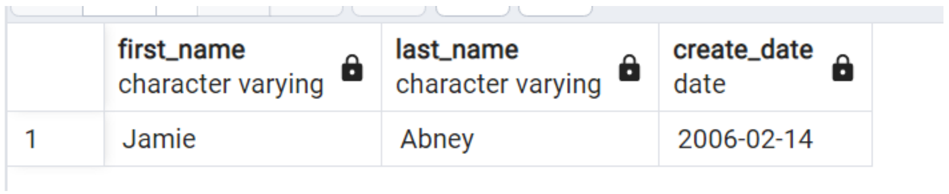
```sql
SELECT  MIN(store_id) AS min_store_id,
        MAX(store_id) AS max_store_id,

        MIN(customer_id) AS min_customer_id,
        MAX(customer_id) AS max_customer_id,

        MIN(address_id) AS min_address_id,
        MAX(address_id) AS max_address_id

FROM customer;
```



min_store_id smallint	max_store_id smallint	min_customer_id integer	max_customer_id integer	min_address_id smallint	max_address_id smallint
1	2	1	599	5	605

Non-numerical

```sql
SELECT  MODE() WITHIN GROUP (ORDER BY first_name) AS first_name,
        MODE() WITHIN GROUP (ORDER BY last_name) AS last_name,
        MODE() WITHIN GROUP (ORDER BY create_date) AS create_date

FROM customer;
```



	first_name character varying	last_name character varying	create_date date
1	Jamie	Abney	2006-02-14

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.

SQL works better when the database is big, it makes it very easy and fast to get the information needed. The important part is to know how to write smartly the script to get fast the information needed.

On the other hand, Excel works very well with small data sets, also if the data is not too big using formulars like highlight duplicate values, or pivot tables is easy to identify anomalies in the data. But if the dataset is too big it gets very challenging and slow to get the information needed.

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