## Juan Ignacio Galvalisi

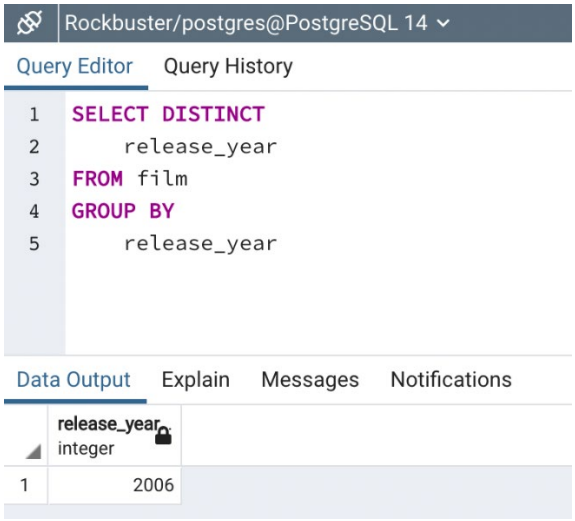
## Exercise 3.6: Summarizing and Cleaning Data in SQL

**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).

### 1. Film table

a. Looking for non-uniform values

Using the DISCTINCT statement we can go through a few random values to check for inconsistencies.




Solution

It could be possible to clean non-uniform data through the UPDATE statement, changing the incorrect values to the correct ones.

## b. Looking for duplicates

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
SELECT
    film_id,
    title,
    description,
    release_year,
    language_id,
    rental_duration,
    rental_rate,
    length,
    replacement_cost,
    rating
FROM film
GROUP BY
    film_id,
    title,
    description,
    release_year,
    language_id,
    rental_duration,
    rental_rate,
    length,
    replacement_cost,
    rating
HAVING COUNT (*) > 1;
```

Data Output | Explain | Messages | Notifications

| film_id [PK] integer | title character varying (255) | description text | release_year integer | language_id smallint | rental_duration smallint | rental_rate numeric (4,2) | length smallint | replacement_cost numeric (5,2) | rating mpaa_rating | last_upd timestam |
|---|---|---|---|---|---|---|---|---|---|---|

## Solution

One solution could be creating a virtual table where you select only unique records. With the CREATE VIEW statement, we are able to create this “View” in order to reject duplicate values.

## c. Looking for missing values

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
1  SELECT
2  *
3  FROM film
4  WHERE
5      film_imdb IS NULL
```

Data Output | Explain | Messages | Notifications

| | film_id [PK] integ | title character va | description text | release_year integer | language_id smallint | rental_duration smallint | rental_rate numeric (4,2) | length smallint | replacement_cost numeric (5,2) | rating mpaa_rating | last_update timestamp without time z | special_features text[] | fulltext tsvector | film_imdb integer |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 133 | Chamber... | A Fateful Refl... | 2006 | 1 | 7 | 4.99 | 117 | 14.99 | NC-17 | 2013-05-26 14:50:58.9... | {Trailers} | 'chamber':1 'fa... | [null] |
| 2 | 384 | Grosse W... | A Epic Drama ... | 2006 | 1 | 5 | 4.99 | 49 | 19.99 | R | 2013-05-26 14:50:58.9... | {"Behind the S... | 'australia':18 'c... | [null] |
| 3 | 8 | Airport P... | A Epic Tale of ... | 2006 | 1 | 6 | 4.99 | 54 | 15.99 | R | 2013-05-26 14:50:58.9... | {Trailers} | 'airport':1 'anci... | [null] |
| 4 | 98 | Bright En... | A Fateful Yarn... | 2006 | 1 | 4 | 4.99 | 73 | 12.99 | PG-13 | 2013-05-26 14:50:58.9... | {Trailers} | 'boat':20 'brigh... | [null] |
| 5 | 1 | Academy... | A Epic Drama ... | 2006 | 1 | 6 | 0.99 | 86 | 20.99 | PG | 2013-05-26 14:50:58.9... | {"Deleted Scen... | 'academi':1 'ba... | [null] |
| 6 | 2 | Ace Gold... | A Astounding ... | 2006 | 1 | 3 | 4.99 | 48 | 12.99 | G | 2013-05-26 14:50:58.9... | {Trailers,"Delet... | 'ace':1 'admini... | [null] |
| 7 | 3 | Adaptati... | A Astounding ... | 2006 | 1 | 7 | 2.99 | 50 | 18.99 | NC-17 | 2013-05-26 14:50:58.9... | {Trailers,"Delet... | 'adapt':1 'astou... | [null] |
| 8 | 4 | Affair Pre... | A Fanciful Do... | 2006 | 1 | 5 | 2.99 | 117 | 26.99 | G | 2013-05-26 14:50:58.9... | {Commentarie... | 'affair':1 'chase... | [null] |
| 9 | 5 | African E... | A Fast-Paced ... | 2006 | 1 | 6 | 2.99 | 130 | 22.99 | G | 2013-05-26 14:50:58.9... | {"Deleted Scen... | 'african':1 'chef... | [null] |
| 10 | 6 | Agent Tr... | A Intrepid Pan... | 2006 | 1 | 3 | 2.99 | 169 | 17.99 | PG | 2013-05-26 14:50:58.9... | {"Deleted Scen... | 'agent':1 'ancie... | [null] |
| 11 | 7 | Airplane ... | A Touching S... | 2006 | 1 | 6 | 4.99 | 62 | 28.99 | PG-13 | 2013-05-26 14:50:58.9... | {Trailers,"Delet... | 'airplan':1 'boat... | [null] |

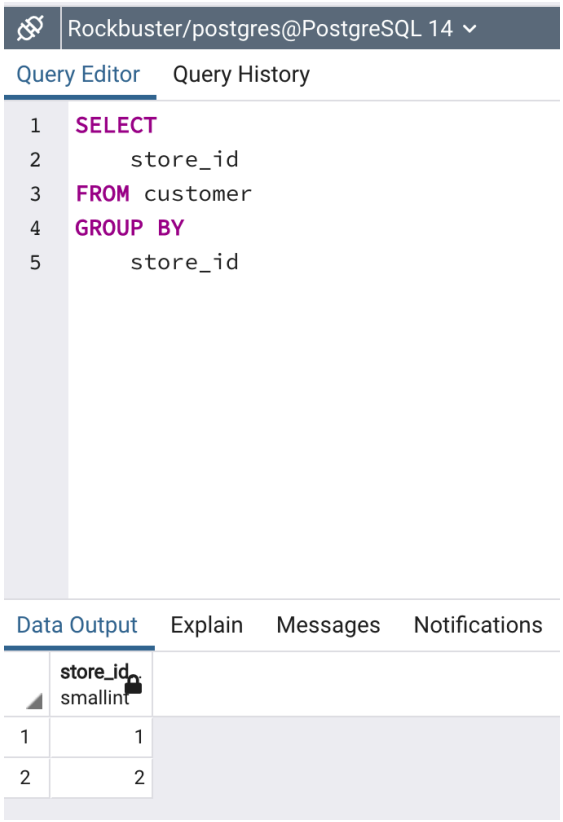
## Solution

Taking into account that the column is completely empty, we could ignore it and exclude it from the following queries, as it is in this case with the film_imdb column.

## 2. Customer table

a. Looking for non-uniform values

Using the GROUP BY statement we can go through a few random values to check for inconsistencies.




Solution

When creating a table, you can use constraints to set the data type for a column. In this sense, you can specify what kind of data a table or a column can accept, but you must perform when a table is created.

b. Looking for duplicates

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
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;
```

Data Output | Explain | Messages | Notifications

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

Solution

One solution could be removing all duplicates. With this, we will keep only values with their unique versions through the DELETE statement.

**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.

## 1. Film table

### a. Numerical values

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
1  SELECT
2      MIN(release_year) AS min_release_year,
3      MAX(release_year) AS max_release_year,
4      ROUND(AVG(release_year),0) AS avg_release_year,
5      MIN(language_id) AS min_language_id,
6      MAX(language_id) AS max_language_id,
7      ROUND(AVG(language_id),2) AS avg_language_id,
8      MIN(rental_duration) AS min_rental_duration,
9      MAX(rental_duration) AS max_rental_duration,
10     ROUND(AVG(rental_duration),2) AS avg_rental_duration,
11     MIN(rental_rate) AS min_rental_rate,
12     MAX(rental_rate) AS max_rental_rate,
13     ROUND(AVG(rental_rate),2) AS avg_rental_rate,
14     MIN(length) AS min_length,
15     MAX(length) AS max_length,
16     ROUND(AVG(length),2) AS avg_length,
17     MIN(replacement_cost) AS min_replacement_cost,
18     MAX(replacement_cost) AS max_replacement_cost,
19     ROUND(AVG(replacement_cost),2) AS avg_replacement_cost
20 FROM film
21
```

Data Output | Explain | Messages | Notifications

| | min_release_year integer | max_release_year integer | avg_release_year numeric | min_language_id smallint | max_language_id smallint | avg_language_id numeric |
|---|---|---|---|---|---|---|
| 1 | 2006 | 2006 | 2006 | 1 | 1 | 1.00 |

| min_rental_duration smallint | max_rental_duration smallint | avg_rental_duration numeric | min_rental_rate numeric | max_rental_rate numeric | avg_rental_rate numeric |
|---|---|---|---|---|---|
| 3 | 7 | 4.99 | 0.99 | 4.99 | 2.98 |

| min_length smallint | max_length smallint | avg_length numeric | min_replacement_cost numeric | max_replacement_cost numeric | avg_replacement_cost numeric |
|---|---|---|---|---|---|
| 46 | 185 | 115.27 | 9.99 | 29.99 | 19.98 |

b. Non-numerical values

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
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,
    mode() WITHIN GROUP (ORDER BY special_features) AS modal_special_features
FROM film;
```

Data Output | Explain | Messages | Notifications

| | modal_title character varying | modal_description text | modal_rating mpaa_rating | modal_special_features text[] |
|---|---|---|---|---|
| 1 | Academy Dinosaur | A Action-Packed Character Study of a Astronaut And a Explo... | PG-13 | {Trailers,Commentaries,"Behind the Scenes"} |

## 2. Customer table

a. Numerical values

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
SELECT
    MIN(customer_id) AS min_customer_id,
    MAX(customer_id) AS max_customer_id,
    ROUND(AVG(customer_id), 2) AS avg_customer_id,
    MIN(store_id) AS min_store_id,
    MAX(store_id) AS max_store_id,
    ROUND(AVG(store_id), 2) AS avg_store_id,
    MIN(address_id) AS min_address_id,
    MAX(address_id) AS max_address_id,
    ROUND(AVG(address_id), 2) AS avg_address_id,
    MIN(active) AS min_active,
    MAX(active) AS max_active,
    ROUND(AVG(active), 2) AS avg_active
FROM customer;
```

Data Output | Explain | Messages | Notifications

| | min_customer_id integer | max_customer_id integer | avg_customer_id numeric | min_store_id smallint | max_store_id smallint | avg_store_id numeric |
|---|---|---|---|---|---|---|
| 1 | 1 | 599 | 300.00 | 1 | 2 | 1.46 |

| min_address_id smallint | max_address_id smallint | avg_address_id numeric | min_active integer | max_active integer | avg_active numeric |
|---|---|---|---|---|---|
| 5 | 605 | 304.72 | 0 | 1 | 0.97 |

b. Non-numerical values

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
1  SELECT
2      mode() WITHIN GROUP (ORDER BY first_name) AS modal_first_name,
3      mode() WITHIN GROUP (ORDER BY last_name) AS modal_last_name,
4      mode() WITHIN GROUP (ORDER BY email) AS modal_email,
5      mode() WITHIN GROUP (ORDER BY first_name) AS modal_first_name
6  FROM customer;
```

Data Output | Explain | Messages | Notifications

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

**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 is operationally faster than Excel when it comes to profiling data, regardless of the size of the dataset. Querying and aliasing functions work better in SQL than in Excel, which translates into time and resource savings. The only difference is that you

need a further step in coding using SQL than Excel, and that could become not so intuitive when making queries.