Databases & SQL for Analysts

3.6: Summarizing & Cleaning Data in SQL

1. Data Cleaning

- **Duplicate data:** no duplicates in either table

```
1.  SELECT title,
2.         description,
3.         release_year,
4.         language_id,
5.  COUNT(*)
6.  FROM public.film
7.  GROUP BY title,
8.           description,
9.           release_year,
10.          language_id
11. HAVING COUNT(*) > 1
```

```
1.  SELECT first_name,
2.         last_name,
3.         email,
4.         address_id,
5.  COUNT(*)
6.  FROM public.customer
7.  GROUP BY first_name,
8.           last_name,
9.           email,
10.          address_id
11. HAVING COUNT(*) > 1
```

If there were duplicates, they would be filtered before analysis by selecting DISTINCT values.

- **Non-uniform data:** 1000/1000 records in film, 599/599 records in customer

```
1. SELECT DISTINCT title,
2.                 description,
3.                 release_year,
4.                 language_id
5. FROM public.film
```

```
1. SELECT DISTINCT first_name,
2.                 last_name,
3.                 email,
4.                 address_id
5. FROM public.customer
```

If there were non-uniform values, they would be updated to the correct ones with UPDATE and SET.

Constraints should be implemented to maintain data uniformity.

- **Incorrect data:**

If possible, to confirm data accuracy, the data should be examined side by side with its sources.

- **Missing data:** no data is missing from either table

```
1. SELECT title,
2.        description,
3.        release_year,
4.        language_id
5. FROM public.film
6. WHERE title IS NULL
7. OR description IS NULL
8. OR release_year IS NULL
9. OR language_id IS NULL
```

```
SELECT first_name,
       last_name,
       email,
       address_id
FROM public.customer
WHERE first_name IS NULL
OR last_name IS NULL
OR email IS NULL
OR address_id IS NULL
```

If there were missing data, depending on the volume, the approach would be to either ignore columns with a lot of missing data or to impute the missing values with the average of the column/the most common value.

2. Data Summary

- **Table film:**

```
SELECT MIN(film_id) AS min_film_id,
       MAX(film_id) AS max_film_id,

       MIN(release_year) AS min_release_year,
       MAX(release_year) AS max_release_year,
       mode() WITHIN GROUP(ORDER BY release_year) AS mode_release_year,

       MIN(language_id) AS min_language_id,
       MAX(language_id) AS max_language_id,
       mode() WITHIN GROUP(ORDER BY language_id) AS mode_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_rent,
       MAX(rental_rate) AS max_rent,
       AVG(rental_rate) AS avg_rent,

```

```
20.        MIN(length) AS min_length,
21.        MAX(length) AS max_length,
22.        AVG(length) AS avg_length,
23.
24.        MIN(replacement_cost) AS min_replacement_cost,
25.        MAX(replacement_cost) AS max_replacement_cost,
26.        AVG(replacement_cost) AS avg_replacement_cost,
27.
28.        mode() WITHIN GROUP(ORDER BY rating) AS mode_rating,
29.
30.        MIN(last_update) AS min_last_update,
31.        MAX(last_update) AS max_last_update,
32.        mode() WITHIN GROUP(ORDER BY last_update) AS mode_last_update,
33.
34.        mode() WITHIN GROUP(ORDER BY special_features) AS mode_special_features
35. FROM public.film
```

- **Table customer:**

```
1.  SELECT MIN(customer_id) AS min_customer_id,
2.         MAX(customer_id) AS max_customer_id,
3.
4.         MIN(store_id) AS min_store_id,
5.         MAX(store_id) AS max_store_id,
6.         mode() WITHIN GROUP(ORDER BY store_id) AS mode_store_id,
7.
8.         mode() WITHIN GROUP(ORDER BY first_name) AS mode_first_name,
9.
10.        mode() WITHIN GROUP(ORDER BY last_name) AS mode_last_name,
11.
12.        MIN(address_id) AS min_address_id,
13.        MAX(address_id) AS max_address_id,
14.        mode() WITHIN GROUP(ORDER BY address_id) AS mode_address_id,
15.
16.        mode() WITHIN GROUP(ORDER BY activebool) AS mode_activebool,
17.
```

```
18.        MIN(create_date) AS min_create_date,
19.        MAX(create_date) AS max_create_date,
20.        mode() WITHIN GROUP(ORDER BY create_date) AS mode_create_date,
21.
22.       MIN(last_update) AS min_last_update,
23.        MAX(last_update) AS max_last_update,
24.        mode() WITHIN GROUP(ORDER BY last_update) AS mode_last_update,
25.
26.        mode() WITHIN GROUP(ORDER BY active) AS mode_active
27. FROM public.customer
```

3. Reflection

In my opinion, Excel is generally more user-friendly and faster for basic data profiling tasks such as sorting, filtering, and summarizing data. However, SQL might be better for larger datasets, as it can process data more quickly than Excel.