3.6: Summarizing & Cleaning Data in SQL

Rockbuster's database engineers have loaded some new data into the database, and your manager has asked you to clean and profile it. Follow the instructions below to complete their request:

- **Check for and clean dirty data**

Missing values

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

  - **Customer table**
  -
```
SELECT customer_id, store_id, first_name, last_name, email, address_id, activebool,
create_date, last_update, COUNT(*)
FROM customer
GROUP BY customer_id, store_id, first_name, last_name, email, address_id,
activebool, create_date, last_update
HAVING COUNT(*) >1;
```

- There are no duplicate values in either the film table or the customer table; when we have duplicate values, we create a virtual table known as a "view" in which we pick only unique entries and then delete the duplicate record from the table or view.
-

Non uniform values

  - **Film table**

```
SELECT DISTINCT film_id, title, description, release_year, language_id,
rental_duration, rental_rate, length, replacement_cost, rating, last_update,
special_features
FROM film;
```

  - **Customer table**

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

- The values in either the film or the customer table are uniform; if they are not, we can fix them with the UPDATE command.

- **Summarize your data**

Film table

```
SELECT MIN(film_id) AS min_film_id,
MAX(film_id) AS max_film_id,
AVG(film_id) AS average_film_id,
MIN(release_year) AS min_release_year,
 MAX(release_year) AS max_release_year,
AVG(release_year) AS average_release_year,
MIN(language_id) AS min_language_id,
```

MAX(language_id) AS max_language_id,
AVG(language_id) AS average_language_id,
MIN(rental_duration) AS min_rental_duration,
MAX(rental_duration) AS max_rental_duration,
AVG(rental_duration) AS average_rental_duration,
MIN(rental_rate) AS min_rental_rate,
MAX(rental_rate) AS max_rental_rate,
AVG(rental_rate) AS average_rental_rate,
MIN (length) AS min_length,
MAX (length) AS max_length,
AVG (length) AS average_length,
MIN (replacement_cost) AS min_replacement_cost,
MAX (replacement_cost) AS max_replacement_cost,
AVG (replacement_cost) AS average_replacement_cost,
MODE () WITHIN GROUP (ORDER BY title) AS mode_title,
MODE () WITHIN GROUP (ORDER BY description) AS mode_description,
MODE () WITHIN GROUP (ORDER BY rating) AS mode_rating,
MODE () WITHIN GROUP (ORDER BY special_features) AS
mode_special_features,
MODE () WITHIN GROUP (ORDER BY fulltext) AS mode_fulltext
FROM film;

Query | Query History

```sql
SELECT MIN(film_id) AS min_film_id,
MAX(film_id) AS max_film_id,
AVG(film_id) AS average_film_id,
MIN(release_year) AS min_release_year,
 MAX(release_year) AS max_release_year,
AVG(release_year) AS average_release_year,
MIN(language_id) AS min_language_id,
MAX(language_id) AS max_language_id,
AVG(language_id) AS average_language_id,
MIN(rental_duration) AS min_rental_duration,
MAX(rental_duration) AS max_rental_duration,
AVG(rental_duration) AS average_rental_duration,
MIN(rental_rate) AS min_rental_rate,
MAX(rental_rate) AS max_rental_rate,
AVG(rental_rate) AS average_rental_rate,
MIN (length) AS min_length,
MAX (length) AS max_length,
AVG (length) AS average_length,
MIN (replacement_cost) AS min_replacement_cost,
MAX (replacement_cost) AS max_replacement_cost,
AVG (replacement_cost) AS average_replacement_cost,
MODE () WITHIN GROUP (ORDER BY title) AS mode_title,
MODE () WITHIN GROUP (ORDER BY description) AS mode_description,
MODE () WITHIN GROUP (ORDER BY rating) AS mode_rating,
MODE () WITHIN GROUP (ORDER BY special_features) AS mode_special_features,
MODE () WITHIN GROUP (ORDER BY fulltext) AS mode_fulltext
FROM film;
```

Customer table

SELECT MIN(customer_id) AS min_customer_id,
MAX(customer_id) AS max_customer_id,
AVG(customer_id) AS average_customer_id,
MIN(store_id) AS min_store_id,
MAX(store_id) AS max_store_id,
AVG(store_id) AS average_store_id,
MIN(address_id) AS min_address_id,
MAX(address_id) AS max_address_id,
AVG(address_id) AS average_address_id,
MIN(active) AS min_active,
MAX(active) AS max_active,
AVG(active) AS average_active,
MODE () WITHIN GROUP (ORDER BY first_name) AS mode_first_name,
MODE () WITHIN GROUP (ORDER BY last_name) AS mode_last_name,
MODE () WITHIN GROUP (ORDER BY email) AS mode_email,
MODE () WITHIN GROUP (ORDER BY activebool) AS mode_activebool,
MODE () WITHIN GROUP (ORDER BY active) AS mode_active
FROM customer;

```
SELECT MIN(customer_id) AS min_customer_id,
MAX(customer_id) AS max_customer_id,
AVG(customer_id) AS average_customer_id,
MIN(store_id) AS min_store_id,
 MAX(store_id) AS max_store_id,
AVG(store_id) AS average_store_id,
MIN(address_id) AS min_address_id,
MAX(address_id) AS max_address_id,
AVG(address_id) AS average_address_id,
MIN(active) AS min_active,
MAX(active) AS max_active,
AVG(active) AS average_active,
MODE () WITHIN GROUP (ORDER BY first_name) AS mode_first_name,
MODE () WITHIN GROUP (ORDER BY last_name) AS mode_last_name,
MODE () WITHIN GROUP (ORDER BY email) AS mode_email,
MODE () WITHIN GROUP (ORDER BY activebool) AS mode_activebool,
MODE () WITHIN GROUP (ORDER BY active) AS mode_active
FROM customer;
```

- **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.

I believe that using SQL for data profiling could be faster and easier, but it demands experience and command knowledge. Excel is simpler to use, but it takes significantly longer. Excel may be better in data profiling for small datasets because of this, but SQL is considerably more effective in handling bigger amounts of data.