

Summarizing & Cleaning Data in SQL

1. Check for and clean dirty data:

The screenshot shows a SQL IDE interface with a query editor and a data output table. The query is designed to find duplicate records in the 'film' table based on several columns.

```
1 SELECT film_id, title, release_year, language_id, rental_duration, rental_rate,  
2 replacement_cost, COUNT (*)  
3 FROM film  
4 GROUP BY film_id, title, release_year, language_id, rental_duration, rental_rate,  
5 replacement_cost  
6 HAVING COUNT (*) >1  
7
```

The data output table has the following columns and data types:

| film_id | title | release_year | language_id | rental_duration | rental_rate | replacement_cost | count |
|--------------|-------------------------|--------------|-------------|-----------------|---------------|------------------|--------|
| [PK] integer | character varying (255) | integer | smallint | smallint | numeric (4,2) | numeric (5,2) | bigint |

The screenshot shows a SQL IDE interface with a query editor and a data output table. The query is designed to find duplicate records in the 'customer' table based on several columns.

```
1 SELECT DISTINCT customer_id, first_name, last_name, email, address_id, active,  
2 COUNT (*)  
3 FROM customer  
4 GROUP BY customer_id, first_name, last_name, email, address_id, active  
5 HAVING COUNT (*) >1  
6  
7
```

The data output table has the following columns and data types:

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

There is no returned duplicate value. Duplicate data can be solved by deleting the records if permissible or using a GROUP BY or DISTINCT to select required data.

2. Summarize your data:

The screenshot shows a PostgreSQL query editor window. The query is a complex aggregation query on the 'film' table. It calculates various statistical measures (min, max, avg, mode) for different attributes. The results are displayed in a table on the right side of the editor.

Query:

```

2  MAX(rental_rate) AS max_rental_rate,
3  AVG(rental_rate) AS avg_rental_rate,
4  MIN(rental_duration) AS min_rental_duration,
5  MAX(rental_duration) AS max_rental_duration,
6  AVG(rental_duration) AS avg_rental_duration,
7  MIN(film_id) AS min_film,
8  MAX(film_id) AS max_film,
9  AVG(film_id) AS avg_film,
10 MIN(language_id) AS min_language,
11 MAX(language_id) AS max_language,
12 AVG(language_id) AS avg_language,
13 MIN(length) AS min_length,
14 MAX(length) AS max_length, AVG(length) AS avg_length,
15 MIN(replacement_cost) AS min_replacement_cost,
16 MAX(replacement_cost) AS max_replacement_cost,
17 AVG(replacement_cost) AS avg_replacement_cost,
18 MODE() WITHIN GROUP (ORDER BY rating) AS rating_value,
19 MODE() WITHIN GROUP (ORDER BY special_features) AS feature_value,
20 MODE() WITHIN GROUP (ORDER BY release_year) AS release_year,
21 MODE() WITHIN GROUP (ORDER BY title) AS title_value,
22 MODE() WITHIN GROUP (ORDER BY fulltext) AS fulltext
23 FROM film
  
```

Data output:

| | min_renatl_rate numeric | max_rental_rate numeric | avg_renatal_ra numeric |
|---|----------------------------|----------------------------|---------------------------|
| 1 | 0.99 | 4.99 | |

| | |
|----------------------|---|
| min_renatl_rate | 0.99 |
| max_rental_rate | 4.99 |
| avg_renatal_rate | 2.98 |
| min_rental_duration | 3 |
| max_rental_duration | 7 |
| avg_rental_duration | 4.985 |
| min_film | 1 |
| max_film | 1000 |
| avg_film | 500.5 |
| min_language | 1 |
| max_language | 1 |
| avg_language | 1 |
| min_length | 46 |
| max_length | 185 |
| avg_length | 115.272 |
| min_replacement_cost | 9.99 |
| max_replacement_cost | 29.99 |
| avg_replacement_cost | 19.984 |
| rating_value | PG-13 |
| feature_value | {Trailers,Commentaries,"Behind the Scenes"} |
| release_year | 2006 |
| title_value | Academy Dinosaur |

The screenshot shows a PostgreSQL query editor window. The query is as follows:

```

1 SELECT MIN(customer_id) AS min_customer_id,
2 MAX(customer_id) AS max_customer_id,
3 AVG(customer_id) AS avg_customer_id,
4 MIN(store_id) AS min_store_id,
5 MAX(store_id) AS max_store_id,
6 AVG(store_id) AS avg_store_id,
7 MIN(address_id) AS min_address_id,
8 MAX(address_id) AS max_address_id,
9 AVG(address_id) AS avg_address_id,
10 MIN(create_date) AS min_create_date,
11 MAX(create_date) AS max_create_date,
12 MODE() WITHIN GROUP (ORDER BY create_date) AS create_date,
13 MIN(last_update) AS min_last_update,
14 MAX(last_update) AS max_last_update,
15 MODE() WITHIN GROUP (ORDER BY last_update) AS last_update,
16 MODE() WITHIN GROUP (ORDER BY first_name) AS first_name,
17 MODE() WITHIN GROUP (ORDER BY last_name) AS last_name,
18 MODE() WITHIN GROUP (ORDER BY email) AS email,
19 MODE() WITHIN GROUP (ORDER BY create_date) AS create_date,
20 MODE() WITHIN GROUP (ORDER BY active) AS mode_active
21 FROM customer;

```

The results are displayed in the 'Data output' tab:

| | min_customer_id integer | max_customer_id integer | avg_customer_id numeric | min_store_id integer | max_store_id integer | avg_store_id numeric | min_address_id integer | max_address_id integer | avg_address_id numeric | min_create_date timestamp | max_create_date timestamp | create_date timestamp | min_last_update timestamp | max_last_update timestamp | last_update timestamp | first_name text | last_name text | email text | create_date timestamp | mode_active boolean |
|---|----------------------------|----------------------------|----------------------------|-------------------------|-------------------------|-------------------------|---------------------------|---------------------------|---------------------------|------------------------------|------------------------------|--------------------------|------------------------------|------------------------------|--------------------------|--------------------|-------------------|--------------------------------|--------------------------|------------------------|
| 1 | 1 | 599 | 300 | 1 | 2 | 1.455759599 | 5 | 605 | 304.7245409 | 14/02/2006 | 14/02/2006 | 14/02/2006 | 49:45.7 | 49:45.7 | 49:45.7 | Jamie | Abney | aaron.selby@sakilacustomer.org | 14/02/2006 | 1 |

| | |
|-----------------|--------------------------------|
| min_customer_id | 1 |
| max_customer_id | 599 |
| avg_customer_id | 300 |
| min_store_id | 1 |
| max_store_id | 2 |
| avg_store_id | 1.455759599 |
| min_address_id | 5 |
| max_address_id | 605 |
| avg_address_id | 304.7245409 |
| min_create_date | 14/02/2006 |
| max_create_date | 14/02/2006 |
| create_date | 14/02/2006 |
| min_last_update | 49:45.7 |
| max_last_update | 49:45.7 |
| last_update | 49:45.7 |
| first_name | Jamie |
| last_name | Abney |
| email | aaron.selby@sakilacustomer.org |
| create_date-2 | 14/02/2006 |
| mode_active | 1 |

3. **Reflect on your work:** Using Excel would be more effective when dealing with small amounts of data, or few columns within a data table to categorize and data profile. However, large amounts of data need SQL to be more effective.