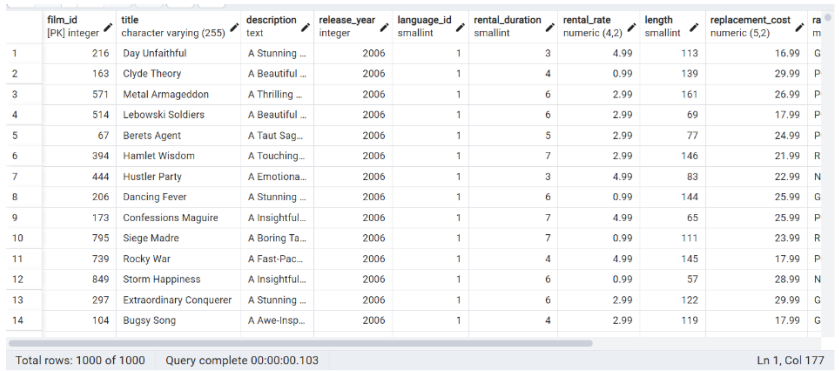
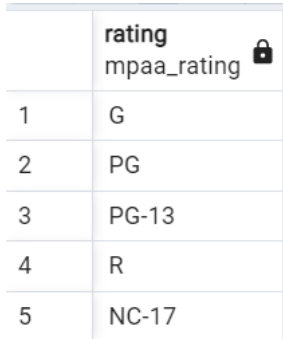


## Task 3.6: Summarizing and Cleaning Data in SQL

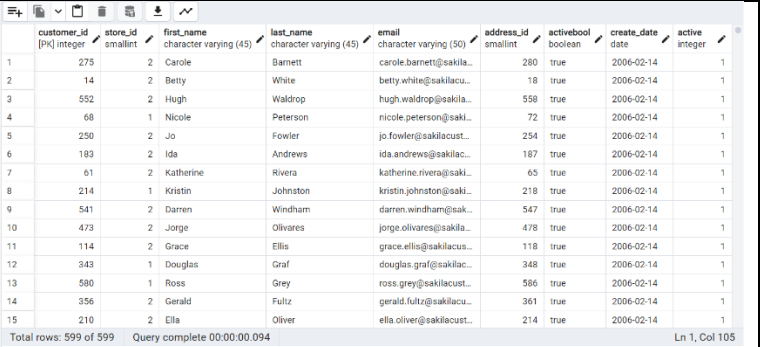
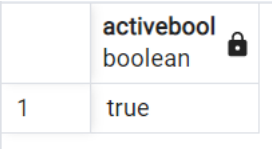
### 1. Check for and clean dirty data.

#### Non-Uniform for Film table:


Query:	Explanation:
<b>SELECT</b> <b>DISTINCT</b> <b>film_id, title,</b> <b>description,</b> <b>release_year,</b> <b>language_id,</b> <b>rental_duration,</b> <b>rental_rate, length,</b> <b>replacement_cost,</b> <b>rating,</b> <b>last_update,</b> <b>special_features</b> <b>FROM film;</b>	<p>1) This shows you whole entire film table and each column within the film table.</p> 
<b>SELECT</b> <b>DISTINCT rating</b> <b>FROM film</b> <b>GROUP BY</b> <b>rating;</b>	<p>2) This shows you the individual column, rating, of the film table. This can be done for each individual column.</p> 

#### Non-Uniform for Customer table:

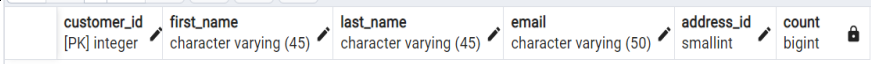
Query:	Explanation:
<b>SELECT DISTINCT</b> <b>customer_id, store_id,</b> <b>first_name, last_name,</b> <b>email, address_id,</b> <b>activebool, create_date,</b> <b>active FROM customer;</b>	<p>1) This shows you whole entire customer table and each column within the customer table.</p>

	
<b>SELECT DISTINCT activebool FROM customer GROUP BY activebool;</b>	<p>2) This shows you the individual column, activebool, of the customer table. This can be done for each individual column.</p> 

Duplicate Data for Film table:

Query:	Result screenshot:
<b>SELECT film_id, title, release_year, language_id, rental_duration,  COUNT(*)  FROM film  GROUP BY film_id, title, release_year, language_id, rental_duration  HAVING COUNT(*) &gt;1;</b>	

Duplicate Data for Customer table:

Query:	Result screenshot:
<pre> SELECT customer_id, first_name, last_name, email, address_id,  COUNT(*)  FROM customer  GROUP BY customer_id, first_name, last_name, email, address_id  HAVING COUNT(*) &gt;1; </pre>	

There are no duplicates for either table (film/customer). If there were duplicates, there are a couple ways to deal with duplicates. You can create a virtual table “View” where unique records can be selected, or duplicate records can be deleted. This would require permission to alter the table. If permission to alter the table is not granted, GROUP BY or DISTINCT functions can be used to select unique records.

### Missing Data:

Queries:	Explanations:
<pre> SELECT column 1, column 3 FROM tablename  --column 2 is ignored in select because of the numerous missing values </pre>	<p>1) If there is a column with too much missing data, it is best to leave it alone as erasing/replacing will cause more problems later.</p>
<pre> UPDATE tablename SET = AVG (column1) WHERE column1 IS NULL </pre>	<p>2) If there are a few missing values, those missing values can be imputed with the average value.</p>

2. Summarize your data.

Descriptive Statistics for Film Table
<b>SELECT</b>  <b>MIN</b> (film_id) <b>AS</b> min_film_id,  <b>MAX</b> (film_id) <b>AS</b> max_film_id,  <b>AVG</b> (film_id) <b>AS</b> avg_film_id,  <b>MIN</b> (release_year) <b>AS</b> min_release_year,  <b>MAX</b> (release_year) <b>AS</b> max_release_year,  <b>AVG</b> (release_year) <b>AS</b> avg_release_year,  <b>MIN</b> (language_id) <b>AS</b> min_language_id,  <b>MAX</b> (language_id) <b>AS</b> max_language_id,  <b>AVG</b> (language_id) <b>AS</b> avg_language_id,  <b>MIN</b> (rental_duration) <b>AS</b> min_rental_duration,  <b>MAX</b> (rental_duration) <b>AS</b> max_rental_duration,  <b>AVG</b> (rental_duration) <b>AS</b> avg_rental_duration,  <b>MIN</b> (rental_rate) <b>AS</b> min_rental_rate,  <b>MAX</b> (rental_rate) <b>AS</b> max_rental_rate,  <b>AVG</b> (rental_rate) <b>AS</b> avg_rental_rate,  <b>MIN</b> (length) <b>AS</b> min_length,  <b>MAX</b> (length) <b>AS</b> max_length,  <b>AVG</b> (length) <b>AS</b> avg_length,  <b>MIN</b> (replacement_cost) <b>AS</b> min_replacement_cost,

```

MAX(replacement_cost) AS max_replacement_cost,

AVG(replacement_cost) AS avg_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;

```

#### Descriptive Statistics for Customer Table

```

SELECT

MIN(customer_id) AS min_customer_id,

MAX(customer_id) AS max_customer_id,

AVG(customer_id) AS avg_customer_id,

MIN(store_id) AS min_store_id,

MAX(store_id) AS max_store_id,

AVG(store_id) AS avg_store_id,

MIN(address_id) AS min_address_id,

MAX(address_id) AS max_address_id,

AVG(address_id) AS avg_address_id,

MIN(create_date) AS min_create_date,

MAX(create_date) AS max_create_date,

```

```
MIN(last_update) AS min_last_update,  
MAX(last_update) AS max_last_update,  
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 active) AS mode_active,  
MODE () WITHIN GROUP (ORDER BY create_date) AS mode_create_date,  
MODE () WITHIN GROUP (ORDER BY last_update) AS mode_last_update  
FROM customer;
```

3. **Reflect on your work:** I think that SQL is more effective for data profiling. With the correct SQL query, the result will be given immediately. However, with Excel there are multiple steps involved in the process of cleaning your data. SQL is easier to use and gives faster results. In addition, SQL is the better choice for large datasets.