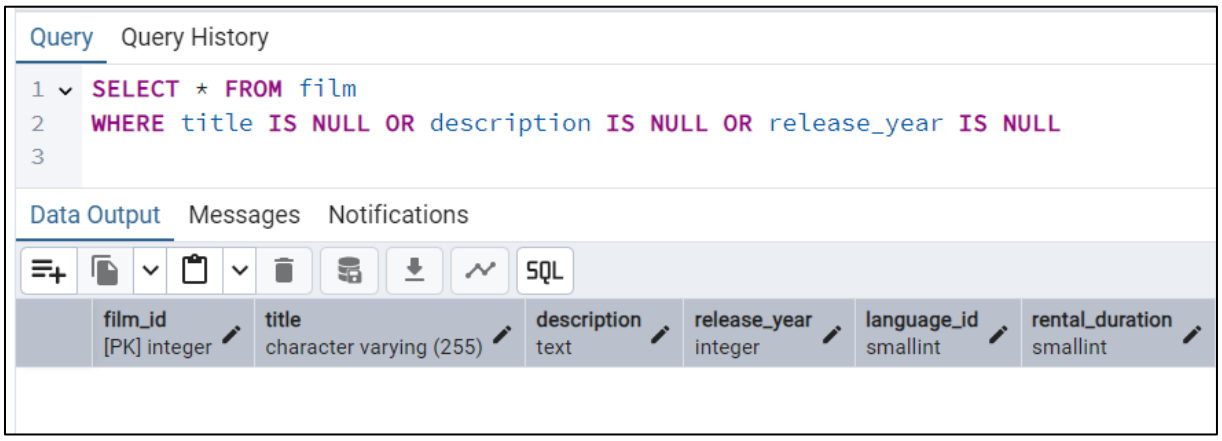
Answers 3.6

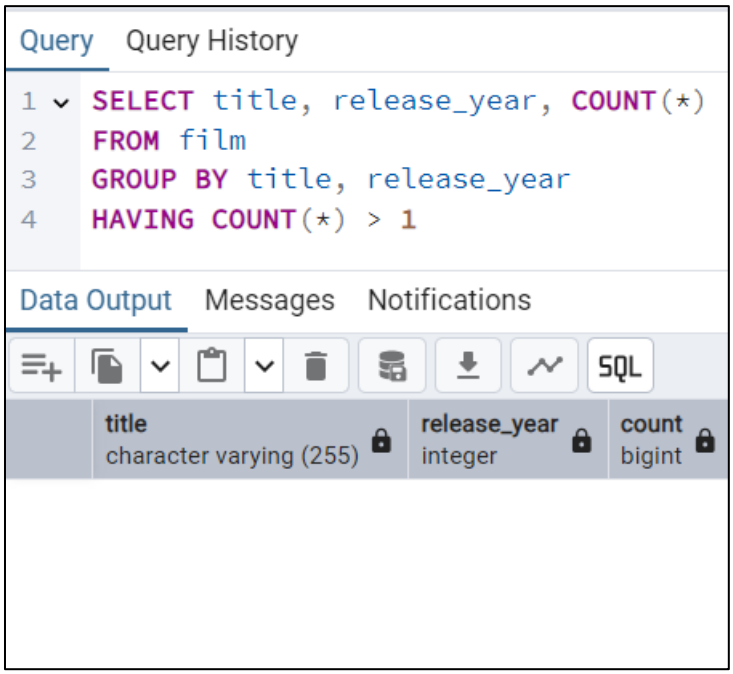
STEP 1:

1. Checking for Missing Values in the film Table:



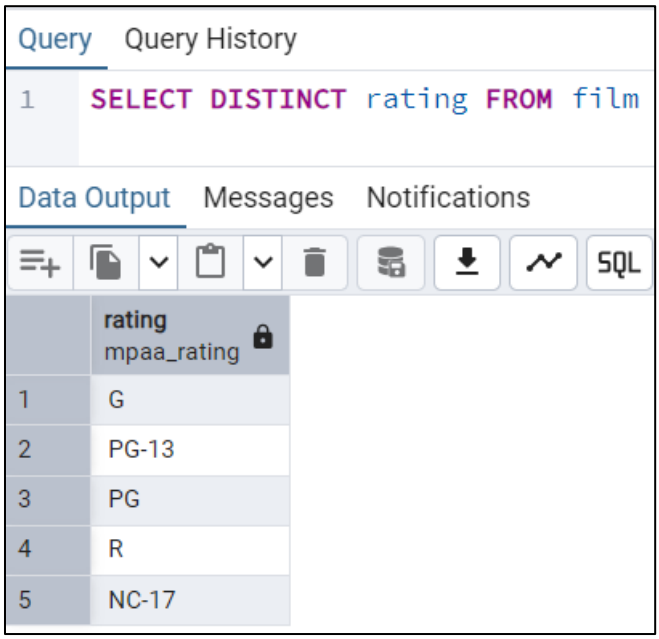
Explanation: This query identifies any rows in the film table where key fields (title, description, or release_year) are missing. If any records are found, we can either remove them if they are incomplete or attempt to fill in missing values from external sources or similar entries.

2. Checking for Duplicates in the film Table:



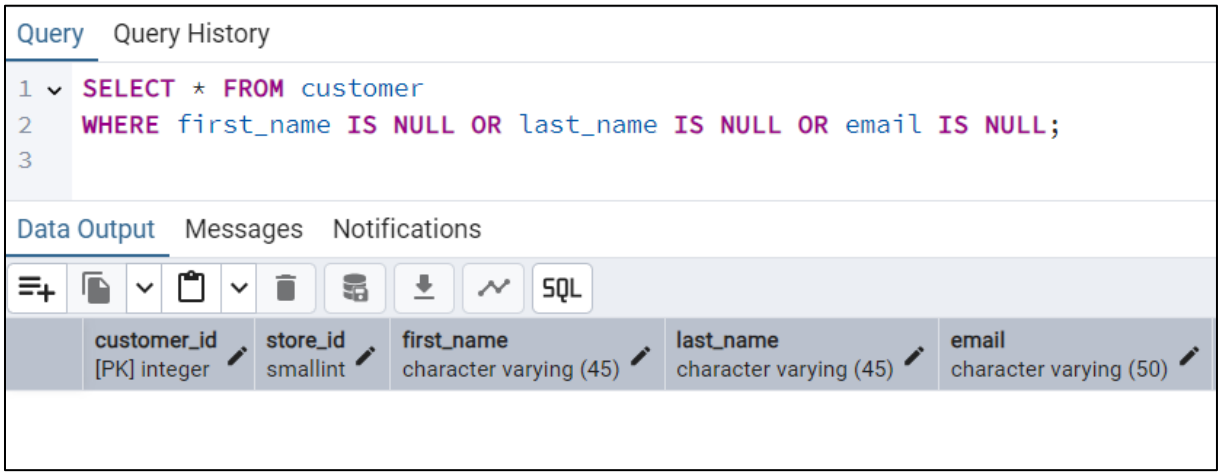
Explanation: This query looks for duplicate films by grouping them based on title and release_year. If duplicates are found, we need to investigate whether they are actual duplicates or legitimate separate entries. If they are unintended duplicates, we may delete the extra rows or merge them.

3. Checking for Non-Uniform Data in the film Table (Rating Column):



Explanation: Since rating is an ENUM type, we expect only the predefined values (G, PG, PG-13, R, NC-17). If any unexpected values appear, they may be due to data entry errors. Cleaning could involve updating incorrect values to match the allowed ENUM values.

4. Checking for Missing Values in the customer Table:

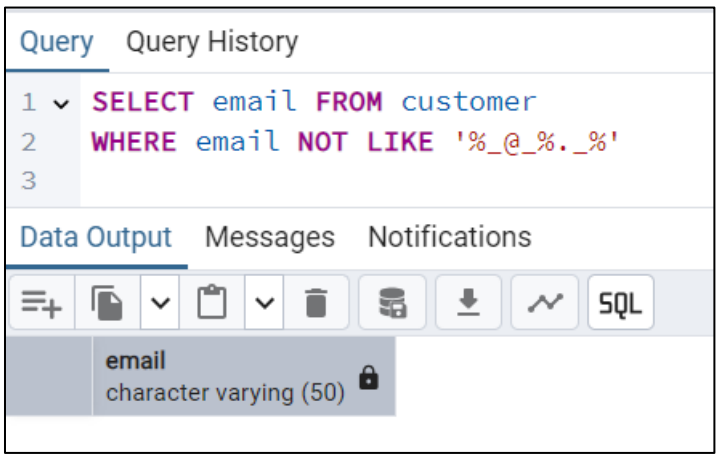


Explanation: This query finds customers with missing names or emails. If found, we can check if the information can be retrieved from other sources or delete the incomplete records if necessary.

5. Checking for Duplicate Customers:

Explanation: This query checks for duplicate customers based on their name and email. If duplicates are found, we can merge records or remove redundant ones.

6. Checking for Inconsistent Data in the customer Table (Email Format):



Explanation: This query identifies improperly formatted email addresses. Cleaning methods include manual corrections or filtering out invalid emails.

STEP 2:

Descriptive Statistics for the film and customer Tables

1. Descriptive Statistics for the film Table:

```
Query  Query History
1  SELECT MIN(rental_duration) AS min_rental_duration,
2         MAX(rental_duration) AS max_rental_duration,
3         AVG(rental_duration) AS avg_rental_duration,
4         MIN(rental_rate) AS min_rental_rate,
5         MAX(rental_rate) AS max_rental_rate,
6         AVG(rental_rate) AS avg_rental_rate,
7         MIN(length) AS min_length,
8         MAX(length) AS max_length,
9         AVG(length) AS avg_length
10 FROM film
```

Data Output Messages Notifications

Showing rows: 1 to 1

	min_rental_duration smallint	max_rental_duration smallint	avg_rental_duration numeric	min_rental_rate numeric	max_rental_rate numeric	avg_rental_rate numeric	min_length smallint	max_length smallint	avg_length numeric
1	3	7	4.9850000000000000	0.99	4.99	2.9800000000000000	46	185	115.2720000000000000

Explanation: This query calculates the minimum, maximum, and average values for numerical columns like rental_duration, rental_rate, and length. These statistics provide insights into the range and distribution of numerical attributes in the film table.

2. Frequent Calculation for the film Table:

```
Query  Query History
1  SELECT rating, COUNT(*) AS count
2  FROM film
3  GROUP BY rating
4  ORDER BY count DESC
5
```

Data Output Messages Notifications

	rating mpaa_rating	count bigint
1	PG-13	223
2	NC-17	210
3	R	195
4	PG	194
5	G	178

Explanation: This query finds the count value in the rating column, which helps us understand the most common film rating in the database.

3. Mode Calculation for the film table:

```
Query  Query History
1  SELECT MODE() WITHIN GROUP (ORDER BY rating)
2  AS modal_vale
3  From film
4
5
```

Data Output | Messages | Notifications

	modal_vale mpaa_rating
1	PG-13

Explanation: This query finds the most frequently occurring (mode) value in the rating column, which helps us understand the most common film rating in the database.

4. Descriptive Statistics for the customer Table:

```
Query  Query History
1  SELECT MIN(create_date) AS earliest_customer,
2         MAX(create_date) AS latest_customer,
3         COUNT(*) AS total_customers
4  FROM customer
5
```

Data Output | Messages | Notifications

	earliest_customer date	latest_customer date	total_customers bigint
1	2006-02-14	2006-02-14	599

Explanation: This query provides the date of the earliest and latest customer entries, along with the total number of customers in the database.

5. Frequent Calculation for the customer Table (Store_id):

Query Query History

```
1 SELECT store_id, COUNT(*) AS count
2 FROM customer
3 GROUP BY store_id
4 ORDER BY count DESC
5
```

Data Output Messages Notifications

	store_id smallint	count bigint
1	1	326
2	2	273

Explanation: This query finds the frequent of the store where customers are registered, giving insight into the geographic distribution of the customer base.

6. Mode Calculation for the customer Table (Store_id):

Query Query History

```
1 SELECT MODE() WITHIN GROUP (ORDER BY store_id)
2 AS modal_value,
3 COUNT(store_id)
4 FROM customer
5 GROUP BY store_id
6 Limit 1
```

Data Output Messages Notifications

	modal_value smallint	count bigint
1	1	326

Explanation: This query finds the most common city where customers are registered, giving insight into the geographic distribution of the customer base.

STEP 3:

Reflection on Data Profiling: Excel vs. SQL

Based on my experience with data profiling in Excel and SQL, SQL is the more effective tool for large datasets and complex queries. SQL allows for automated, scalable, and efficient data analysis, making it ideal for profiling structured databases. It can quickly handle large volumes of data, perform aggregations, and enforce constraints to ensure data integrity. In contrast, Excel is more user-friendly and visually intuitive but is less efficient for large datasets and lacks the power of SQL in handling relational data and running complex queries. While Excel is useful for quick analysis and visualization, SQL is the superior tool for comprehensive data profiling in enterprise environments.