

## 1. Check for and clean dirty data

### Film table

#### 1. Duplicate data

The screenshot shows a SQL query editor with a query to find duplicate records in the film table. The query is as follows:

```
1 SELECT title, release_year, language_id, rental_duration, COUNT(*)
2 FROM film
3 GROUP BY title, release_year, language_id, rental_duration
4 HAVING COUNT(*) > 1
```

The query is executed, and the results are shown in the Data Output tab. The results are as follows:

title	release_year	language_id	rental_duration	count
character varying (255)	integer	smallint	smallint	bigint

No duplicates were found in the data. There are two ways to fix duplicates.

- Create a view table where we will select only unique records.
- Delete the duplicate records from the table or view.

#### 2. Non-Uniform data

The screenshot shows a SQL query editor with a query to find non-uniform data in the film table. The query is as follows:

```
1 SELECT rating
2 FROM film
3 GROUP BY rating
```

The query is executed, and the results are shown in the Data Output tab. The results are as follows:

rating
mpaa_rating
G
PG-13
PG
R
NC-17

There is no non-uniform data found. To fix non-uniform data, I will use UPDATE command to set the rating consistent.

### 3. Missing data

The screenshot shows a SQL IDE with a query editor on the left and a data output pane at the bottom. The query is as follows:

```
1 SELECT *
2 FROM film
3 WHERE film_id IS NULL
4 OR title IS NULL
5 OR description IS NULL
6 OR release_year IS NULL
7 OR language_id IS NULL
8 OR rental_duration IS NULL
9 OR rental_rate IS NULL
10 OR length IS NULL
11 OR replacement_cost IS NULL
12 OR rating IS NULL
13 OR last_update IS NULL
14 OR special_features IS NULL
15 OR fulltext IS NULL
```

The data output pane at the bottom shows the schema of the film table:

film_id	title	description	release_year	language_id	rental_duration	rental_rate	length	replacement_cost	rating	last_update	special_features
[PK] integer	character varying (255)	text	integer	smallint	smallint	numeric (4,2)	smallint	numeric (5,2)	mpaa_rating	timestamp without time zone	text[]

No missing data was found. If there is data missing then we can simply ignore the columns with the high percentage of missing values. Another option is to impute the values using statistical methods. We can input the estimate or average values.

### Customer table

#### 1. Duplicate data

The screenshot shows a SQL IDE with a query editor on the left and a data output pane at the bottom. The query is as follows:

```
1 SELECT customer_id, store_id, first_name, last_name, email, COUNT(*)
2 FROM customer
3 GROUP BY customer_id, store_id, first_name, last_name, email
4 HAVING COUNT(*) > 1
```

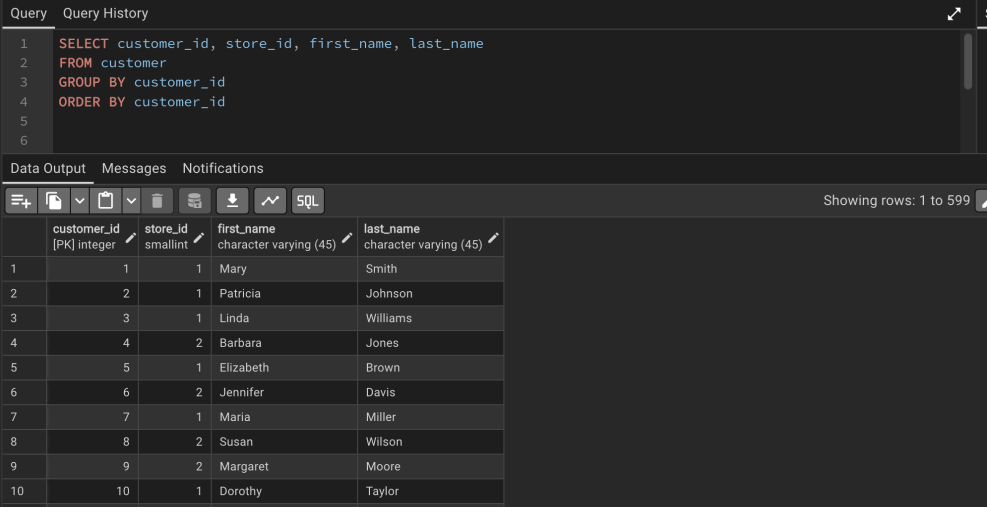
The data output pane at the bottom shows the schema of the customer table:

customer_id	store_id	first_name	last_name	email	count
[PK] integer	smallint	character varying (45)	character varying (45)	character varying (50)	bigint

No duplicates were found in the data. There are two ways to fix duplicates.

- Create a view table where we will select only unique records.
- Delete the duplicate records from the table or view.

## 2. Non-Uniform data



The screenshot shows a SQL query editor with a query window and a data output window. The query is as follows:

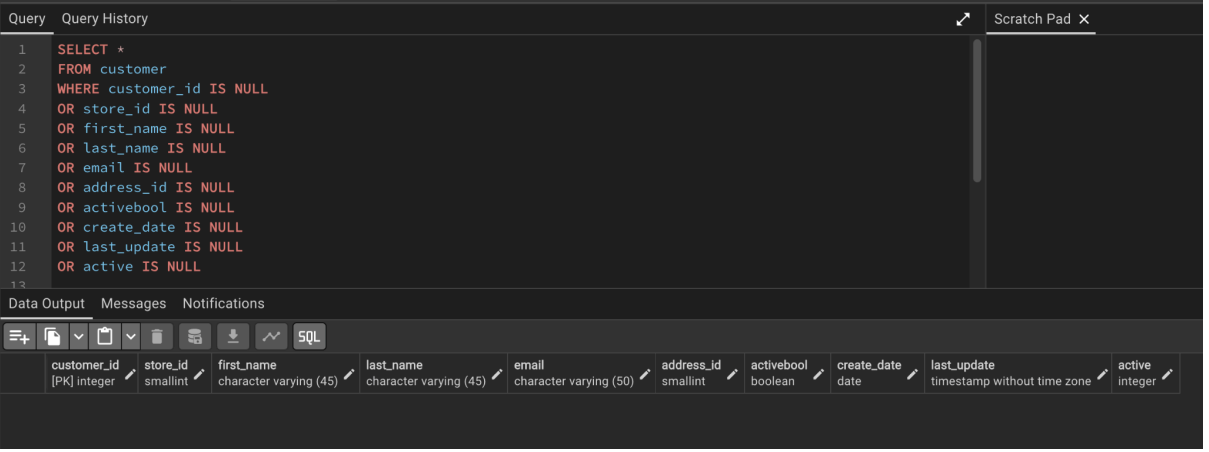
```
1 SELECT customer_id, store_id, first_name, last_name
2 FROM customer
3 GROUP BY customer_id
4 ORDER BY customer_id
5
6
```

The data output window shows the results of the query, displaying 10 rows of data. The columns are customer\_id, store\_id, first\_name, and last\_name. The data is grouped by customer\_id, showing that each customer has a unique store\_id and first\_name, but some customers have multiple last\_names.

customer_id	store_id	first_name	last_name
1	1	Mary	Smith
2	2	Patricia	Johnson
3	1	Linda	Williams
4	4	Barbara	Jones
5	5	Elizabeth	Brown
6	6	Jennifer	Davis
7	7	Maria	Miller
8	8	Susan	Wilson
9	9	Margaret	Moore
10	10	Dorothy	Taylor

No non-uniform data found. To fix non-uniform data, i will use GROUP BY or DISTINCT commands.

## 3. Missing data



The screenshot shows a SQL query editor with a query window and a data output window. The query is as follows:

```
1 SELECT *
2 FROM customer
3 WHERE customer_id IS NULL
4 OR store_id IS NULL
5 OR first_name IS NULL
6 OR last_name IS NULL
7 OR email IS NULL
8 OR address_id IS NULL
9 OR activebool IS NULL
10 OR create_date IS NULL
11 OR last_update IS NULL
12 OR active IS NULL
13
```

The data output window shows the results of the query, displaying 10 rows of data. The columns are customer\_id, store\_id, first\_name, last\_name, email, address\_id, activebool, create\_date, last\_update, and active. The data is filtered to show only rows where any of the specified columns are NULL.

customer_id	store_id	first_name	last_name	email	address_id	activebool	create_date	last_update	active
1	1	Mary	Smith						
2	2	Patricia	Johnson						
3	1	Linda	Williams						
4	4	Barbara	Jones						
5	5	Elizabeth	Brown						
6	6	Jennifer	Davis						
7	7	Maria	Miller						
8	8	Susan	Wilson						
9	9	Margaret	Moore						
10	10	Dorothy	Taylor						

No missing data was found. If there is data missing then we can simply ignore the columns with the high percentage of missing values. Another option is to impute the values using statistical methods. We can input the estimate or average values.

## 2. Summarize your data

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

SELECT MIN(rental\_rate) AS min\_rate,

MAX(rental\_rate) AS max\_rate,

AVG(rental\_rate) AS avg\_rate,

COUNT(rental\_rate) AS count\_values,

COUNT(\*) AS count\_rows

FROM film

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

📥

📡

SQL

min\_rate

numeric

max\_rate

numeric

avg\_rate

numeric

count\_values

bigint

count\_rows

bigint

1

0.99

4.99

2.9800000000000000

1000

1000

Query	Query History
1	SELECT MODE() WITHIN GROUP (ORDER BY title) AS modal_value
2	FROM film
3	
4	
5	
6	
-	

Data Output	Messages	Notifications
<div><div>≡+</div><div>📄</div><div>▼</div><div>📋</div><div>▼</div><div>🗑️</div><div>🗄️</div><div>📥</div><div>📡</div><div>SQL</div></div>		
	modal_value character varying	
1	Academy Dinosaur	

### Film table

	MIN	MAX	AVG	COUNT	COUNT_ROWS	MODE
film_id	1	1000	500.50	1000	1000	
title				1000	1000	Academy Dinosaur
description				1000	1000	
release_year	2006	2006	2006	1000	1000	

language_id	1	1	1	1000	1000	
Rental_duration	3	7	4.99	1000	1000	
length	46	185	115.27	1000	1000	
replacement_cost	9.99	29.99	19.98	1000	1000	
rating				1000	1000	PG-13
last_update				1000	1000	2013-05-26
special_feature				1000	1000	Trailers, Commentaries
fulltext				1000	1000	Balloon 19 confront

Customer table

	MIN	MAX	AVG	COUNT	COUNT_Rows	MODE
customer_id	1	599	300	599	599	
store_id	1	2	1.455	599	599	
first_name				599	599	Jamie
email				599	599	Aaron.selby@sakilacustomer.org
address_id	5	605	304.72	599	599	
activebool				599	599	true
create_date				599	599	2006-02-14
last_update				599	599	2013-05-26
active	0	1	0.97	599	599	

### 3. **Reflect on work**

Between Excel and SQL, SQL is more effective for data profiling. SQL can handle very large datasets quickly and lets you write queries to check for missing values, duplicates, or unusual patterns without slowing down. It is also more reliable because the results come directly from the database. Excel, on the other hand, is easier to use for small datasets and quick visual checks, but it becomes slow and hard to manage when the data is large or complex. Overall, SQL is faster, more powerful, and better for professional data profiling, while Excel is good for small or simple checks.