

1. Check for and clean dirty data

Film table

1. Duplicate data

The screenshot shows a SQL query interface with a query editor and a results table. The query is:

```
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 results table has the following columns and data:

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 interface with a query editor and a results table. The query is:

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

The results table has the following columns and data:

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 query editor interface. The top bar has tabs for 'Query' and 'Scratch Pad'. The main area contains the following SQL code:

```
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
```

Below the code, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The bottom part of the interface shows the schema for the 'film' table:

film_id	[PK] integer	title	character varying (255)	description	text	release_year	integer	language_id	smallint	rental_duration	smallint	rental_rate	numeric (4,2)	length	smallint	replacement_cost	numeric (5,2)	rating	mpaa_rating	last_update	timestamp without time zone	special_features	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 query editor interface. The top bar has tabs for 'Query' and 'Scratch Pad'. The main area contains the following SQL code:

```
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
```

Below the code, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The bottom part of the interface shows the schema for the 'customer' table:

customer_id	[PK] integer	store_id	smallint	first_name	character varying (45)	last_name	character varying (45)	email	character varying (50)	count	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 interface with a dark theme. The top bar has tabs for 'Query' (selected), 'Query History', and 'Scratch Pad'. Below the tabs is a toolbar with icons for file operations and a 'SQL' button. The main area contains a code editor with the following SQL query:

```
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
```

Below the code editor is a table titled 'Data Output' showing the results of the query. The table has four columns: 'customer_id' (integer, PK), 'store_id' (smallint), 'first_name' (character varying(45)), and 'last_name' (character varying(45)). The data shows 10 rows of sample customer data.

customer_id	store_id	first_name	last_name
1	1	Mary	Smith
2	2	Patricia	Johnson
3	3	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

Text at the bottom right of the interface says 'Showing rows: 1 to 599'.

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 interface with a dark theme. The top bar has tabs for 'Query' (selected), 'Query History', and 'Scratch Pad'. Below the tabs is a toolbar with icons for file operations and a 'SQL' button. The main area contains a code editor with the following SQL query:

```
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
```

Below the code editor is a table titled 'Data Output' showing the results of the query. The table has 12 columns corresponding to the fields checked in the WHERE clause: customer_id, store_id, first_name, last_name, email, address_id, activebool, create_date, last_update, and active. All columns are defined as smallint, character varying(45), character varying(50), smallint, boolean, date, timestamp without time zone, and integer respectively. The data table is currently empty.

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

The screenshot shows a SQL query interface with the following details:

Query History:

```
1 SELECT MIN(rental_rate) AS min_rate,
2       MAX(rental_rate) AS max_rate,
3       AVG(rental_rate) AS avg_rate,
4       COUNT(rental_rate) AS count_values,
5       COUNT(*) AS count_rows
6 FROM film
7
8
9
10
11
12
13
```

Data Output:

	min_rate	max_rate	avg_rate	count_values	count_rows
1	0.99	4.99	2.980000000000000	1000	1000

The screenshot shows a SQL query interface with the following details:

Query History:

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

Data Output:

	modal_value
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, Comments
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.