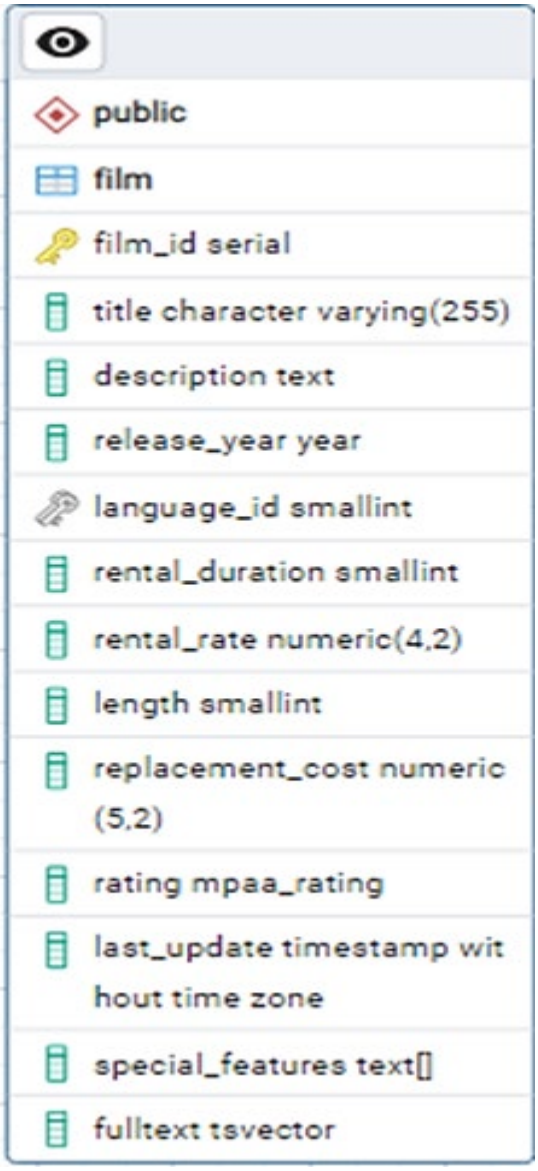
2024

Answers 3.6

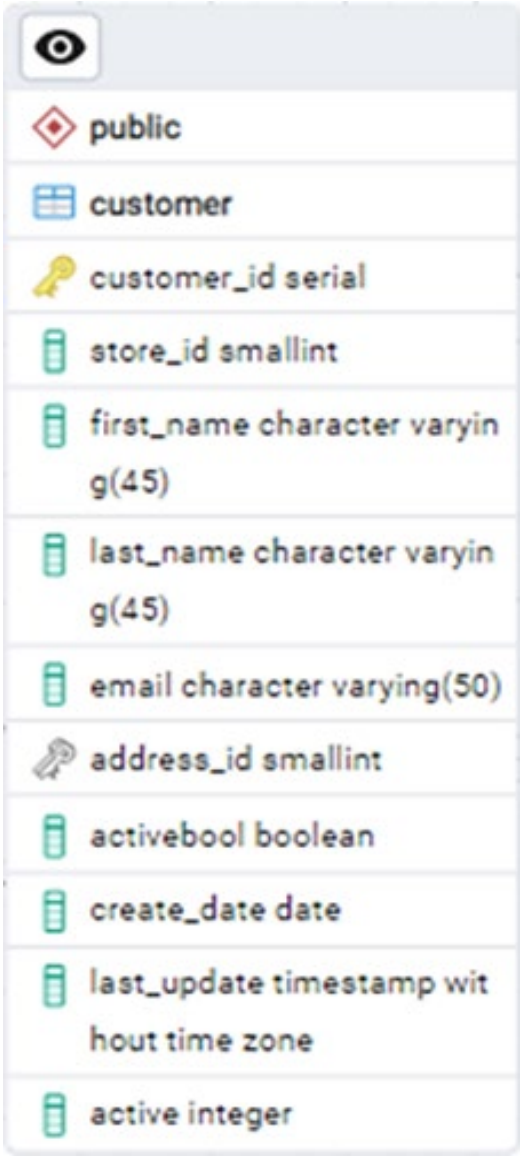
IVONNE ASPILCUETA

1. **Check for and clean dirty data:** Find out if the film table and the customer table contain any dirty data:

Film Table



Customer Table



A. NON-UNIFORM VALUES

QUERY FILM TABLE USING GROUP BY:

```
SELECT film_id,
        title,
        description,
        release_year,
       language_id,
       rental_duration,
       rental_rate,
       length,
       replacement_cost,
      rating,
      special_features
FROM film
GROUP BY film_id,
        title,
        description,
        release_year,
        language_id,
       rental_duration,
      rental_rate,
      length,
      replacement_cost,
      rating,
      special_features;
```

OUTPUT

1000 records from 1000 records were returned, meaning all records are unique.

Another approach for Text values

```
SELECT
      rating,
      special_features,
      fulltext
FROM film
WHERE rating NOT IN ('G', 'PG', 'PG-13', 'R',
'NC-17');
```

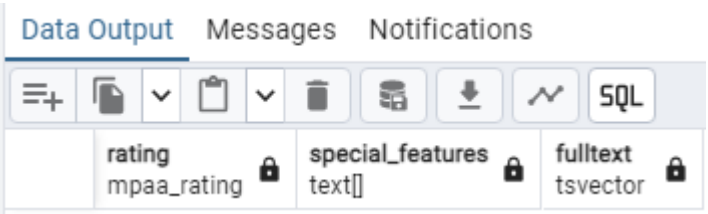
QUERY CUSTOMER TABLE USING DISTINCT:

```
SELECT DISTINCT
        customer_id,
        store_id,
        first_name,
        last_name,
        email,
        address_id,
        activebool,
        create_date,
        active
FROM customer;
```

OUTPUT

All 599 records from 599 records were returned, No non-uniform were found.

OUTPUT



How would you clean the data?

Non-uniform values in the database can be clean using the **UPDATE** statement:

B. DUPLICATE VALUES

QUERY FILM TABLE

```
SELECT
        film_id,
        title,
        description,
        release_year,
        language_id,
        rental_duration,
        rental_rate,
        length,
        replacement_cost,
        rating,
        special_features,
COUNT(*)
FROM film
GROUP BY
        film_id,
        title,
        description,
        release_year,
        language_id,
        rental_duration,
        rental_rate,
        length,
        replacement_cost,
        rating,
        special_features
HAVING COUNT(*) >1;
```

QUERY CUSTOMER TABLE

```
SELECT
        customer_id,
        store_id,
        first_name,
        last_name,
        email,address_id,
        activebool,
        create_date,
        active,
COUNT(*)
FROM customer
GROUP BY
        customer_id,
        store_id,
        first_name,
        last_name,
        email,address_id,
        activebool,
        create_date,
        active
HAVING COUNT(*) >1;
```

OUTPUT

No duplicates found.

OUTPUT
No duplicates found.

How would you clean the data?

There are two ways to clean the data:

1. Create a virtual table, known as a "view," where you select only unique records.
2. Delete the duplicate record from the view table.

C. **MISSING DATA**

QUERY FILM TABLE

```
SELECT *
FROM film
WHERE film_id IS NULL
      OR title IS NULL
      OR description IS NULL
      OR release_year IS NULL
      OR language_id IS NULL
      OR rental_duration IS NULL
      OR rental_rate IS NULL
      OR length IS NULL
      OR replacement_cost IS NULL
      OR rating IS NULL
      OR special_features IS NULL;
```

OUTPUT
No missing data found.

QUERY CUSTOMER TABLE

```
SELECT *
FROM customer
WHERE customer_id IS NULL
      OR store_id IS NULL
      OR first_name IS NULL
      OR last_name IS NULL
      OR Email IS NULL
      OR address_id IS NULL
      OR activebool IS NULL
      OR create_date IS NULL
      OR active IS NULL;
```

OUTPUT
No missing data found.

How would you clean the data?

- Simply ignore columns with a high percentage of missing values. To do so in SQL, you simply omit whichever column you want to ignore from your SELECT statement:

```
SELECT col1,
    col2,
    col4... --col3 ignored in select because it has a lot of missing values
FROM tablename
```

- I**mpute values** is to "fill in" missing values with estimates. With Query:

```
 UPDATE tablename
 SET = AVG(col1)
 WHERE col1 IS NULL
```

- Delete records if missing data is <5%.

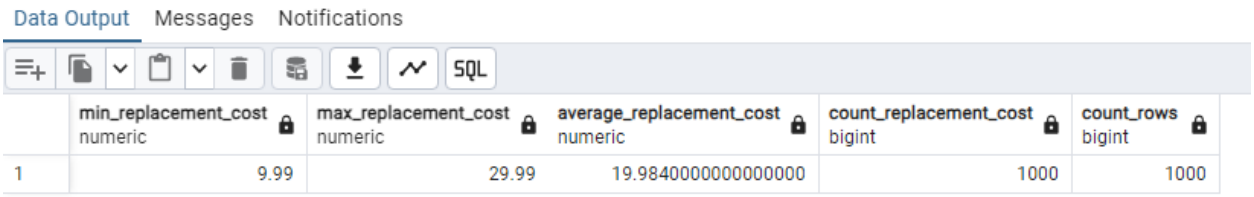
2. **Summarize your data:** Use SQL to calculate descriptive statistics for both the film table and the customer table.

   A. FILM TABLE
   Numeric columns: release_year, rental_duration, rental_rate, length, replacement_cost
   Queries:

```
SELECT MIN(rental_duration) AS min_rental_duration,
        MAX(rental_duration) AS max_rental_duration,
        AVG(rental_duration) AS average_rental_duration,
        COUNT(rental_duration) AS count_rental_duration,
        COUNT(*) AS count_rows
FROM film;

SELECT MIN(release_year) AS min_ release_year,
        MAX(release_year) AS max_ release_year,
        AVG(release_year) AS average_ release_year,
        COUNT(release_year) AS count_ release_year,
        COUNT(*) AS count_rows
FROM film;


SELECT MIN(rental_rate) AS min_rental_rate,
        MAX(rental_rate) AS max_rental_rate,
        AVG(rental_rate) AS average_rental_rate,
        COUNT(rental_rate) AS count_rental_rate,
        COUNT(*) AS count_rows
FROM film;


SELECT MIN(length) AS min_film_length,
        MAX(length) AS max_film_length,
        AVG(length) AS average_film_length,
        COUNT(length) AS count_film_length,
         COUNT(*) AS count_rows
FROM film;
```

```
SELECT MIN(replacement_cost) AS min_replacement_cost,
       MAX(replacement_cost) AS max_replacement_cost,
       AVG(replacement_cost) AS average_replacement_cost,
       COUNT(replacement_cost) AS count_replacement_cost,
        COUNT(*) AS count_rows
FROM film;
```

Data Output Example:



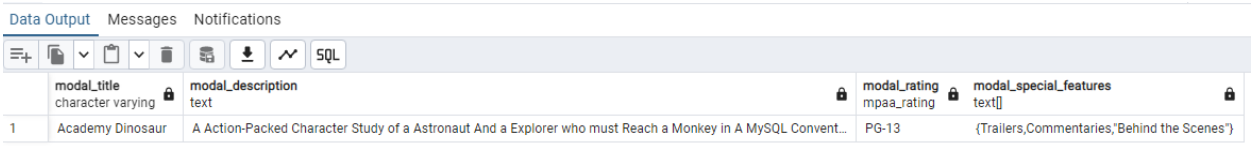
Data Output | Messages | Notifications

	min_replacement_cost numeric	max_replacement_cost numeric	average_replacement_cost numeric	count_replacement_cost bigint	count_rows bigint
1	9.99	29.99	19.9840000000000000	1000	1000

Non-Numerical Variables: title, description, rating, special_features
Query:

```
SELECT
MODE() WITHIN GROUP (ORDER BY title) AS modal_title,
MODE() WITHIN GROUP (ORDER BY description) AS modal_description,
MODE() WITHIN GROUP (ORDER BY rating) AS modal_rating,
MODE() WITHIN GROUP (ORDER BY special_features) AS modal_special_features
FROM film;
```

Data Output:



Data Output | Messages | Notifications

	modal_title character varying	modal_description text	modal_rating mpaa_rating	modal_special_features text[]
1	Academy Dinosaur	A Action-Packed Character Study of a Astronaut And a Explorer who must Reach a Monkey in A MySQL Convent...	PG-13	{Trailers,Commentaries,"Behind the Scenes"}

B. CUSTOMER TABLE

Numerical Variables: store_id, address_id, active

```
SELECT MIN(store_id) AS min_store_id,
       MAX(store_id) AS max_store_id,
       AVG(store_id) AS average_store_id,
       COUNT(store_id) AS count_store_id,
       COUNT(*) AS count_rows
FROM customer;
```

```
SELECT MIN(address_id) AS min_address_id,
	MAX(address_id) AS max_address_id,
	AVG(address_id) AS average_address_id,
	COUNT(address_id) AS count_address_id,
	COUNT(*) AS count_rows
FROM customer;
```

```
SELECT MIN(active) AS min_active,
	MAX(active) AS max_active,
	AVG(active) AS average_active,
	COUNT(active) AS count_active,
	COUNT(*) AS count_rows
FROM customer;
```

Data Output Example:

Data Output | Messages | Notifications

	min_store_id smallint	max_store_id smallint	average_store_id numeric	count_store_id bigint	count_rows bigint
1	1	2	1.4557595993322204	599	599

Non-Numerical Variables: first_name, last_name, email

```
SELECT MODE() WITHIN GROUP (ORDER BY first_name) AS modal_first_name,
	MODE() WITHIN GROUP (ORDER BY last_name) AS modal_last_name,
	MODE() WITHIN GROUP (ORDER BY email) AS modal_email
FROM customer;
```

Output Data:

Data Output | Messages | Notifications

	modal_first_name character varying	modal_last_name character varying	modal_email character varying
1	Jamie	Abney	aaron.selby@sakilacustomer.org

3. **Reflect on your work:** Back in Achievement 1 you learned about data profiling in Excel. Based on your previous experience, which tool (Excel or SQL) do you think is more effective for data profiling, and why? Consider their respective functions, ease of use, and speed. Write a short paragraph in the running document that you have started.

   By all means, profiling with SQL is more effective and accurate, Excel involves manual process can take more time than profiling in SQL.

   When working to find duplicates, missing values, etc. it is easier and faster to work in SQL, the structure is cleaner to work on it.

   EXCEL is good to find patterns, by visualizing the data or creating PIVOT tables.