

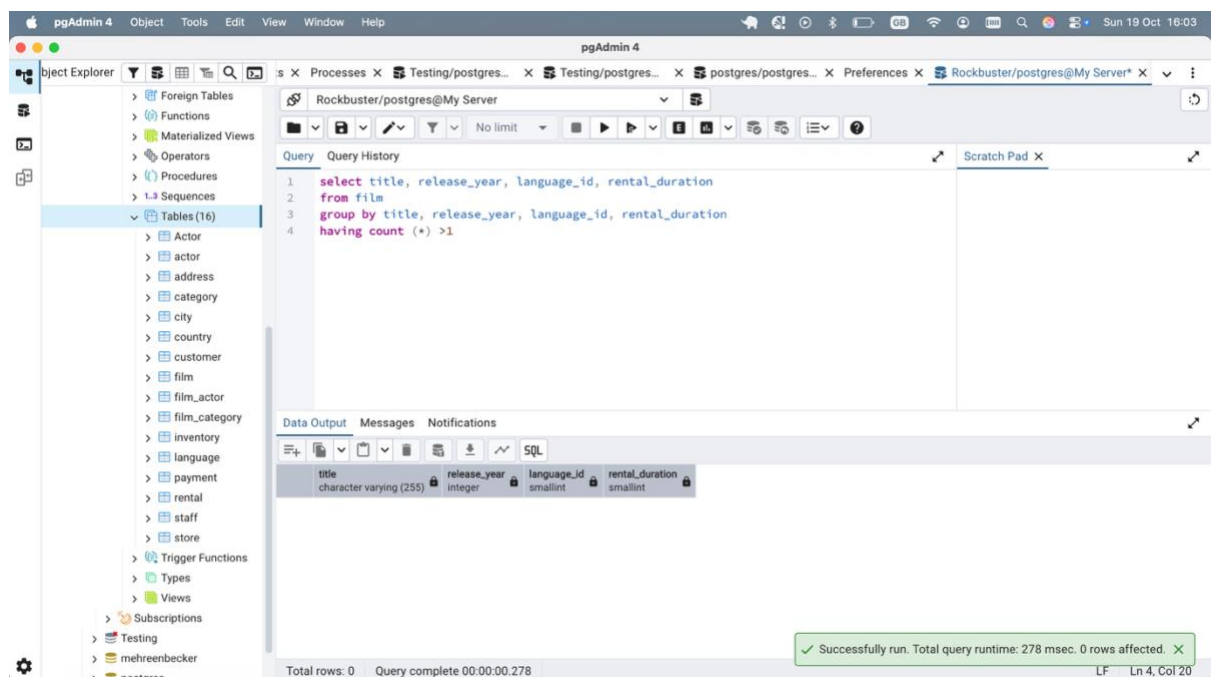
Data Analytics Immersion

3.6: Summarizing & Cleaning Data in SQL

Mehreen Becker

Rockbuster's database engineers have loaded some new data into the database, and your manager has asked you to clean and profile it. Follow the instructions below to complete their request:

1. Check for and clean dirty data: Find out if the film table and the customer table contain any dirty data, specifically non-uniform or duplicate data, or missing values. Create a new "Answers 3.6" document and copy-paste your queries into it. Next to each query write 2 to 3 sentences explaining how you would clean the data (even if the data is not dirty)



I would first identify duplicate records by analyzing key fields (SELECT column, COUNT (*) FROM table GROUP BY column HAVING COUNT (*) >1) to see where the same values appear more than once and then I would decide which records to keep and remove the extra duplicates to ensure only unique and accurate data remains.

pgAdmin 4 Object Tools Edit View Window Help

pgAdmin 4

Rockbuster/postgres@My Server

Query Query History

```
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
```

Data Output Messages Notifications

| film_id | title | description | release_year | language_id | rental_duration | rental_rate | length | replacement_cost | rating | last_update |
|--------------|-------------------------|-------------|--------------|-------------|-----------------|---------------|----------|------------------|-------------|-------------|
| [PK] integer | character varying (255) | text | integer | smallint | smallint | numeric (4,2) | smallint | numeric (5,2) | mpaa_rating | timestamp |

Total rows: 0 Query complete 00:00:00.090

pgAdmin 4 Object Tools Edit View Window Help

pgAdmin 4

Rockbuster/postgres@My Server

Query Query History

```
1 SELECT rental_duration FROM film GROUP BY rental_duration
```

Data Output Messages Notifications

| rental_duration |
|-----------------|
| 4 |
| 6 |
| 7 |
| 3 |
| 5 |

Showing rows: 1 to 5 Page No: 1 of 1

Successfully run. Total query runtime: 265 msec. 5 rows affected.

Total rows: 5 Query complete 00:00:00.295

pgAdmin 4 Object Tools Edit View Window Help

pgAdmin 4

Rockbuster/postgres@My Server

Query Query History

```
1 SELECT release_year FROM film GROUP BY release_year
2
```

Scratch Pad X

Data Output Messages Notifications

| release_year | integer |
|--------------|---------|
| 1 | 2006 |

Showing rows: 1 to 1 Page No: 1 of 1

Total rows: 1 Query complete 00:00:00.078 LF Ln 1, Col 52

pgAdmin 4 Object Tools Edit View Window Help

pgAdmin 4

Rockbuster/postgres@My Server

Query Query History

```
1 SELECT release_year FROM film GROUP BY release_year
2
3 SELECT rental_rate FROM film GROUP BY rental_rate
4
5 SELECT rental_duration FROM film GROUP BY rental_duration
```

Scratch Pad X

Data Output Messages Notifications

| rental_rate | numeric (4,2) |
|-------------|---------------|
| 1 | 2.99 |
| 2 | 4.99 |
| 3 | 0.99 |

Showing rows: 1 to 3 Page No: 1 of 1

Successfully run. Total query runtime: 73 msec. 3 rows affected.

Total rows: 3 Query complete 00:00:00.073 LF Ln 3, Col 50

Customer Table

pgAdmin 4 Object Tools Edit View Window Help

pgAdmin 4

Rockbuster/postgres@My Server

Query Query History

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

Data Output Messages Notifications

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

Successfully run. Total query runtime: 75 msec. 0 rows affected.

Total rows: 0 Query complete 00:00:00.075

pgAdmin 4 Object Tools Edit View Window Help

pgAdmin 4

Rockbuster/postgres@My Server

Query Query History

```
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
```

Data Output Messages Notifications

| customer_id | store_id | first_name | last_name | email | address_id | activebool | create_date | last_update |
|--------------|----------|------------------------|------------------------|------------------------|------------|------------|-------------|-----------------------------|
| [PK] integer | smallint | character varying (45) | character varying (45) | character varying (50) | smallint | boolean | date | timestamp without time zone |

Successfully run. Total query runtime: 278 msec. 0 rows affected.

Total rows: 0 Query complete 00:00:00.278

pgAdmin 4 interface showing a SQL query and its output. The query is:

```

1 SELECT customer_id
2 FROM customer
3 GROUP BY customer_id

```

The output shows the first 9 rows of the customer table, grouped by customer_id. The status bar indicates 'Total rows: 599' and 'Query complete 00:00:00.228'.

| customer_id [PK] integer |
|--------------------------|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |

pgAdmin 4

ObjectToolsEditViewWindowHelp

Object Explorer

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (16)

Actor

actor

address

category

city

country

customer

film

film_actor

film_category

inventory

language

payment

rental

staff

store

Trigger Functions

Types

Views

Subscriptions

Testing

mehreenbecker

postgres

pgAdmin 4

ProcessesTesting/postgres...Testing/postgres...postgres/postgres...PreferencesRockbuster/postgres@My Server*

Rockbuster/postgres@My Server

Query History

SELECT customer_id, store_id, first_name, last_name, activebool, last_update

FROM customer

GROUP BY customer_id

Scratch Pad

Data OutputMessagesNotifications

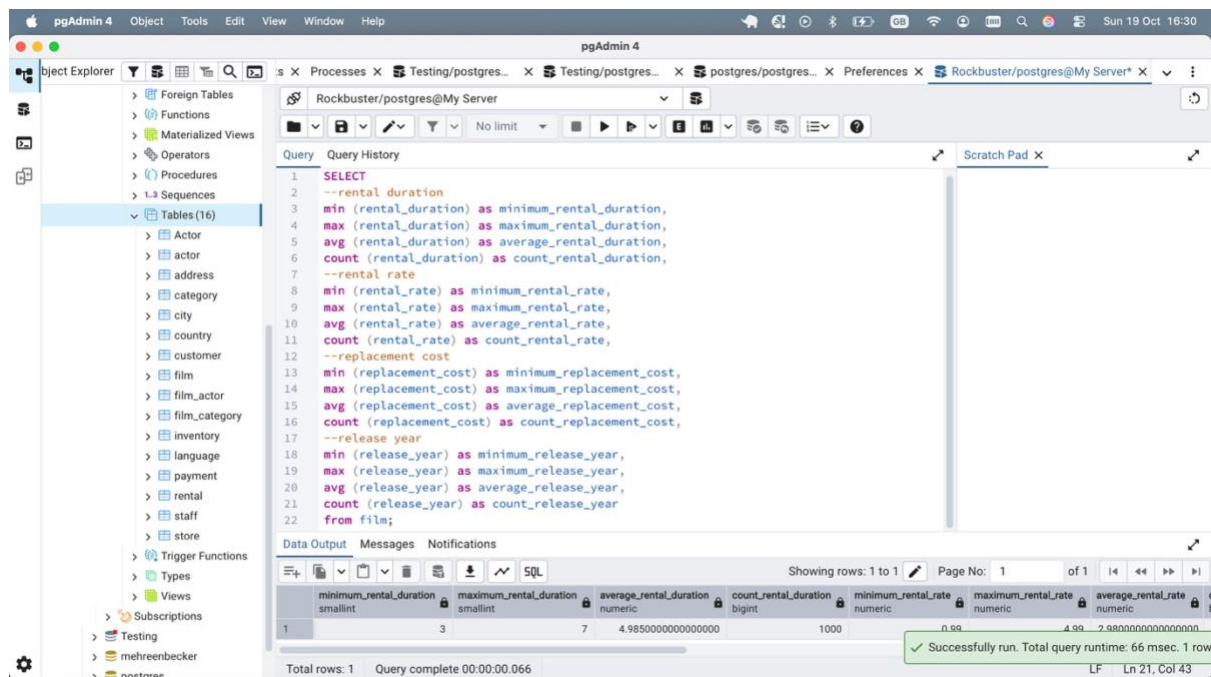
Showing rows: 1 to 599Page No: 1 of 1

| | customer_id [PK] integer | store_id smallint | first_name character varying (45) | last_name character varying (45) | activebool boolean | last_update timestamp without time zone |
|---|--------------------------|-------------------|-----------------------------------|----------------------------------|--------------------|---|
| 1 | 184 | 1 | Vivian | Ruiz | true | 2013-05-26 14:49:45.738 |
| 2 | 87 | 1 | Wanda | Patterson | true | 2013-05-26 14:49:45.738 |
| 3 | 477 | 1 | Dan | Paine | true | 2013-05-26 14:49:45.738 |
| 4 | 273 | 2 | Priscilla | Lowe | true | 2013-05-26 14:49:45.738 |
| 5 | 550 | 2 | Guy | Brownlee | true | 2013-05-26 14:49:45.738 |
| 6 | 394 | 2 | Chris | Brothers | true | 2013-05-26 14:49:45.738 |
| 7 | 51 | 1 | Alice | Stewart | true | 2013-05-26 14:49:45.738 |
| 8 | 272 | 1 | Kay | Caldwell | true | 2013-05-26 14:49:45.738 |
| 9 | 70 | 2 | Christina | Ramirez | true | 2013-05-26 14:49:45.738 |

Total rows: 599Query complete 00:00:00.093

Successfully run. Total query runtime: 93 msec. 599 rows affected.

2. Summarize your data: Use SQL to calculate descriptive statistics for both the film table and the customer table. For numerical columns, this means finding the minimum, maximum, and average values. For non-numerical columns, calculate the mode value. Copy-paste your SQL queries and their outputs into your answers document.



Rental Duration

| minimum_rental_duration | maximum_rental_duration | average_rental_duration | count_rental_duration |
|-------------------------|-------------------------|-------------------------|-----------------------|
| 3 | 7 | 4.9850000000000000 | 1000 |

Rental Rate

| minimum_rental_rate | maximum_rental_rate | average_rental_rate | count_rental_rate |
|---------------------|---------------------|---------------------|-------------------|
| 0.99 | 4.99 | 2.9800000000000000 | 1000 |

Replacement Cost

| minimum_replacement_cost | maximum_replacement_cost | average_replacement_cost | count_replacement_cost |
|--------------------------|--------------------------|--------------------------|------------------------|
| 9.99 | 29.99 | 19.9840000000000000 | 1000 |

Releas Year

| minimum_release_year | maximum_release_year | average_release_year | count_release_year |
|----------------------|----------------------|-----------------------|--------------------|
| 2006 | 2006 | 2006.0000000000000000 | 1000 |

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.

From my experience, as someone who has been working with SQL for about a week, I think Excel feels more straightforward for basic data profiling since it's easy to filter, sort and use quick functions on (like average etc) without having to write out the queries. However, SQL feels more powerful and is generally faster when it comes to handling

larger datasets. I'm sure that when I learn more and am more familiar with the commands, I will be able to profile data more efficiently.