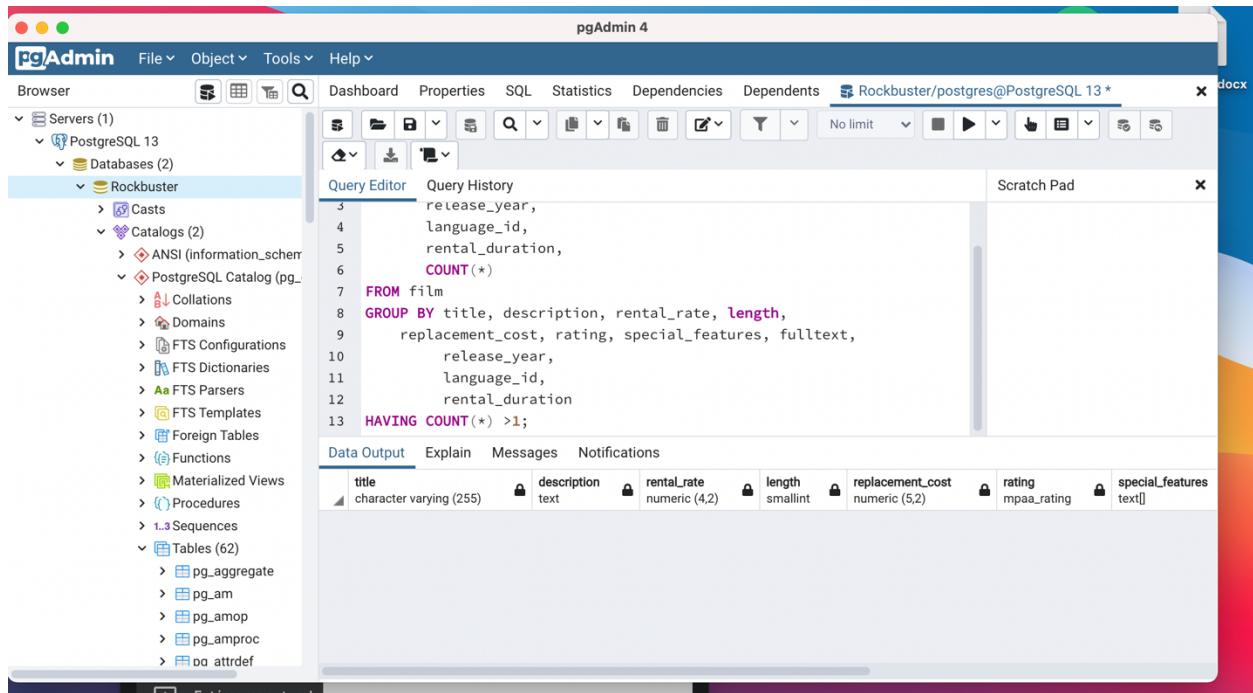


3.6 SUMMARIZING AND CLEANING DATA IN SQL TASK

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).



The screenshot shows the pgAdmin 4 application interface. The left sidebar displays a tree view of the database structure under 'Servers (1) > PostgreSQL 13 > Databases (2) > Rockbuster'. The 'Tables (62)' node is expanded, showing various system tables like pg_aggregate, pg_am, pg_amop, pg_amproc, and pg_attrdef. The main area contains a 'Query Editor' tab with the following SQL query:

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

Below the query editor is a 'Data Output' tab showing the schema for the film table:

title	description	rentalRate	length	replacement_cost	rating	special_features
character varying (255)	text	numeric (4,2)	smallint	numeric (5,2)	mpaa_rating	text[]

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (3) > Rockbuster', several objects are listed: Casts, Catalogs (2), and PostgreSQL Catalog (pg_). In the 'Query Editor' tab, the following SQL query is run:

```
1 SELECT *
2 FROM film
3 WHERE film_id = 1001
```

The results are displayed in a table:

description	release_year	language_id	rental_duration	rental_rate	length	replacement_cost	rating	last_update	special_features
American Marvel Comic Movie	2019	1	3	4.99	[null]	19.99	G	2021-08-1...	[null]

Found No duplicates in Customer

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (2) > Rockbuster', the 'Tables (62)' section is expanded, showing various tables like pg_aggregate, pg_am, pg_amop, pg_amproc, and pg_attrdef.

In the 'Query Editor' tab, the following SQL query is run:

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

The results are displayed in a table:

customer_id	store_id	first_name	last_name	email	address_id	active
[PK] integer	smallint	character varying (45)	character varying (45)	character varying (50)	smallint	boolean

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (2) > Rockbuster', the 'Tables' node is selected. In the 'Query Editor' tab, the following SQL query is entered:

```

1 SELECT *
2 FROM customer
3 WHERE active IS NULL

```

The 'Data Output' tab shows the schema for the 'customer' table:

customer_id	store_id	first_name	last_name	email	address_id	active
[PK] integer	smallint	character varying (45)	character varying (45)	character varying (50)	smallint	boolean

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.

FILM TABLE

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (2) > Rockbuster', the 'Tables' node is selected. In the 'Query Editor' tab, the following SQL query is entered:

```

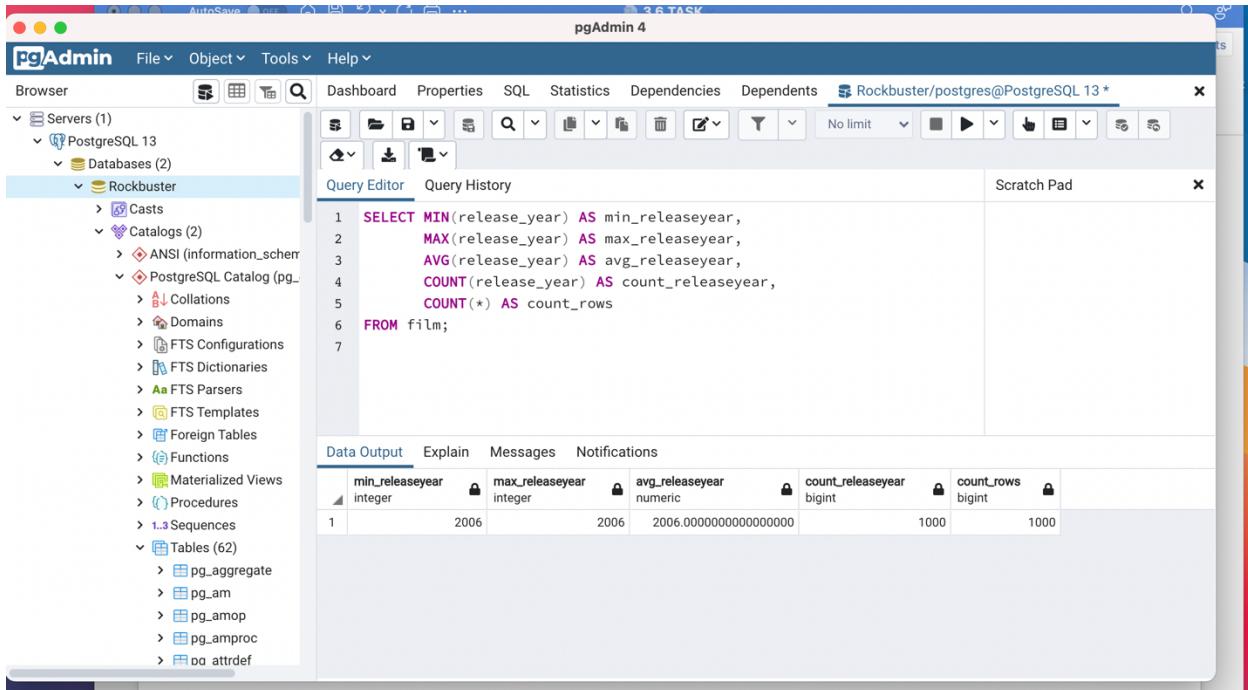
1 SELECT MIN(rental_rate) AS min_rent,
2      MAX(rental_rate) AS max_rent,
3      AVG(rental_rate) AS avg_rent,
4      COUNT(rental_rate) AS count_rent_values,
5      COUNT(*) AS count_rows
6 FROM film;
7

```

The 'Data Output' tab shows the results of the query:

min_rent	max_rent	avg_rent	count_rent_values	count_rows
0.99	4.99	2.980000000000000	1000	1000

BY RELEASE YEAR



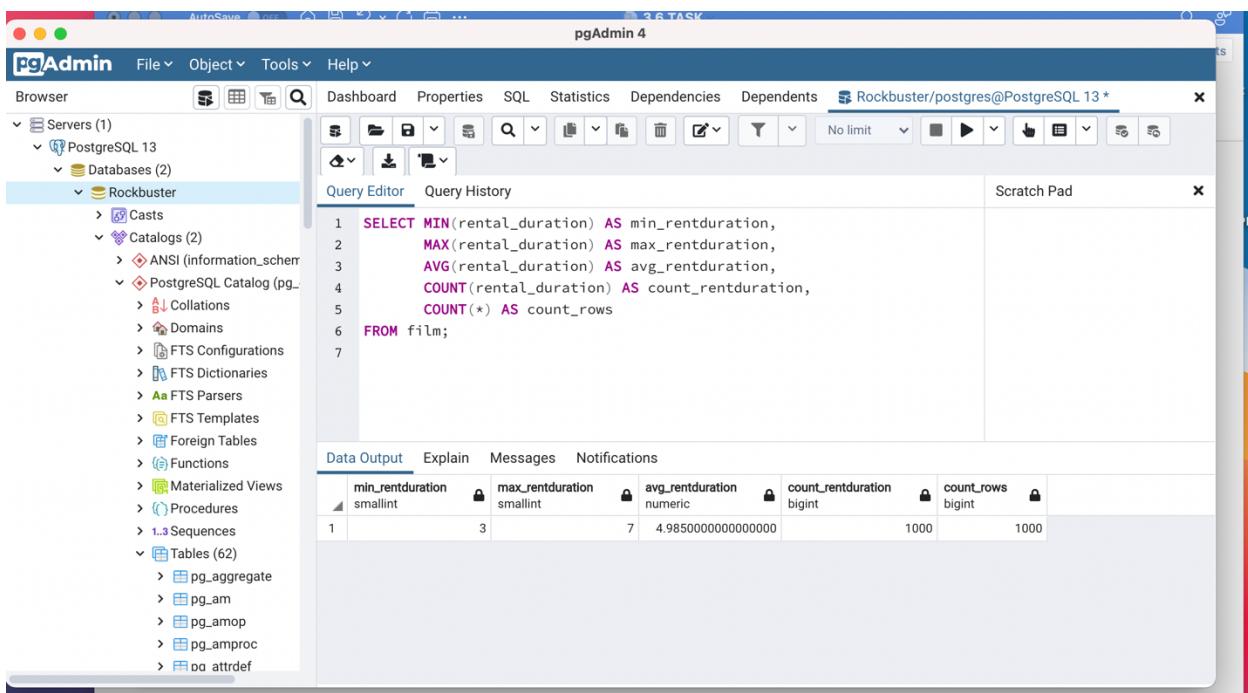
The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Rockbuster' database, including 'Tables (62)' which contains several system tables starting with 'pg_'. The main area shows a SQL query in the 'Query Editor' tab:

```
1 SELECT MIN(release_year) AS min_releaseyear,
2        MAX(release_year) AS max_releaseyear,
3        AVG(release_year) AS avg_releaseyear,
4        COUNT(release_year) AS count_releaseyear,
5        COUNT(*) AS count_rows
6 FROM film;
7
```

The 'Data Output' tab shows the results of the query:

	min_releaseyear	max_releaseyear	avg_releaseyear	count_releaseyear	count_rows
1	2006	2006	2006.0000000000000000	1000	1000

BY RENTAL DURATION



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Rockbuster' database, including 'Tables (62)' which contains several system tables starting with 'pg_'. The main area shows a SQL query in the 'Query Editor' tab:

```
1 SELECT MIN(rental_duration) AS min_renduration,
2        MAX(rental_duration) AS max_renduration,
3        AVG(rental_duration) AS avg_renduration,
4        COUNT(rental_duration) AS count_renduration,
5        COUNT(*) AS count_rows
6 FROM film;
7
```

The 'Data Output' tab shows the results of the query:

	min_renduration	max_renduration	avg_renduration	count_renduration	count_rows
1	3	7	4.985000000000000	1000	1000

BY RENTAL RATE

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Rockbuster' database, including 'Casts', 'Catalogs', 'PostgreSQL Catalog', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', and 'Tables (62)'. The 'Tables' node is expanded to show 'pg_aggregate', 'pg_am', 'pg_amop', 'pg_amproc', and 'oo_attrdef'. The main area shows a query editor with the following SQL code:

```
1 SELECT MIN(rental_rate) AS min_rentalrate,
2        MAX(rental_rate) AS max_rentalrate,
3        AVG(rental_rate) AS avg_rentalrate,
4        COUNT(rental_rate) AS count_rentduration,
5        COUNT(*) AS count_rows
6 FROM film;
7
```

The 'Data Output' tab shows the results of the query:

	min_rentalrate	max_rentalrate	avg_rentalrate	count_rentduration	count_rows
1	0.99	4.99	2.980000000000000	1000	1000

BY LENGTH

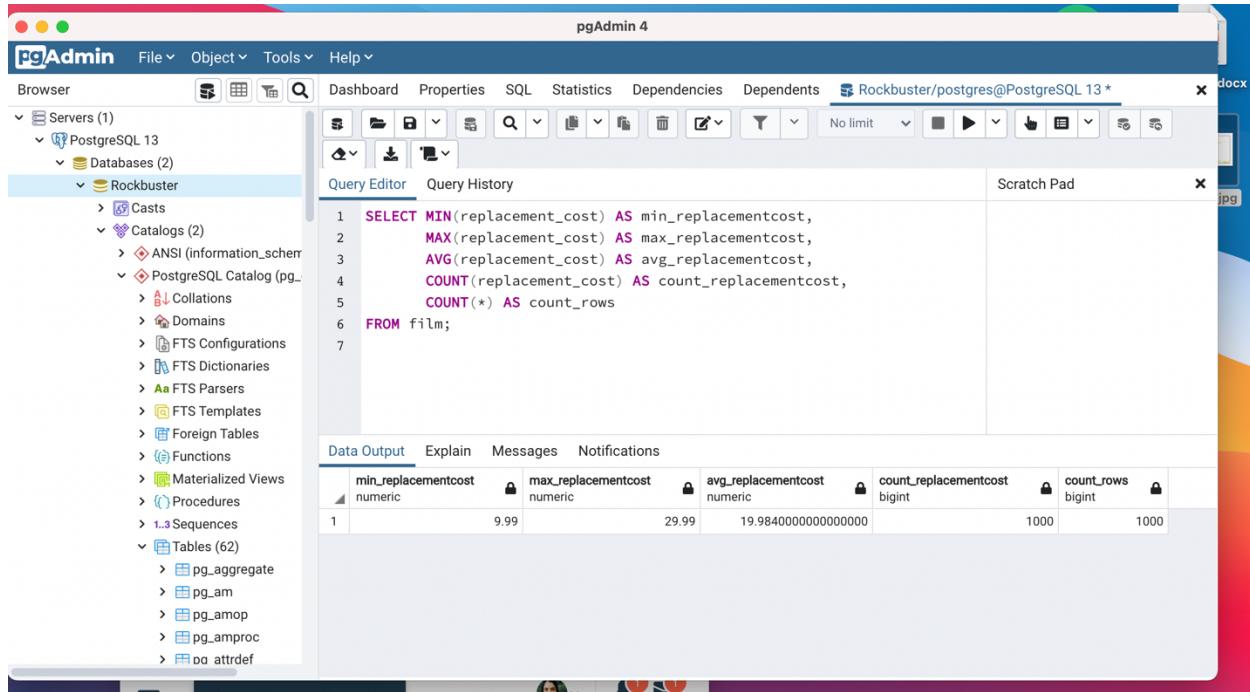
The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Rockbuster' database, including 'Casts', 'Catalogs', 'PostgreSQL Catalog', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', and 'Tables (62)'. The 'Tables' node is expanded to show 'pg_aggregate', 'pg_am', 'pg_amop', 'pg_amproc', and 'oo_attrdef'. The main area shows a query editor with the following SQL code:

```
1 SELECT MIN(length) AS min_length,
2        MAX(length) AS max_length,
3        AVG(length) AS avg_length,
4        COUNT(length) AS count_length,
5        COUNT(*) AS count_rows
6 FROM film;
7
```

The 'Data Output' tab shows the results of the query:

	min_length	max_length	avg_length	count_length	count_rows
1	46	185	000000000000	1000	1000

REPLACEMENT COST



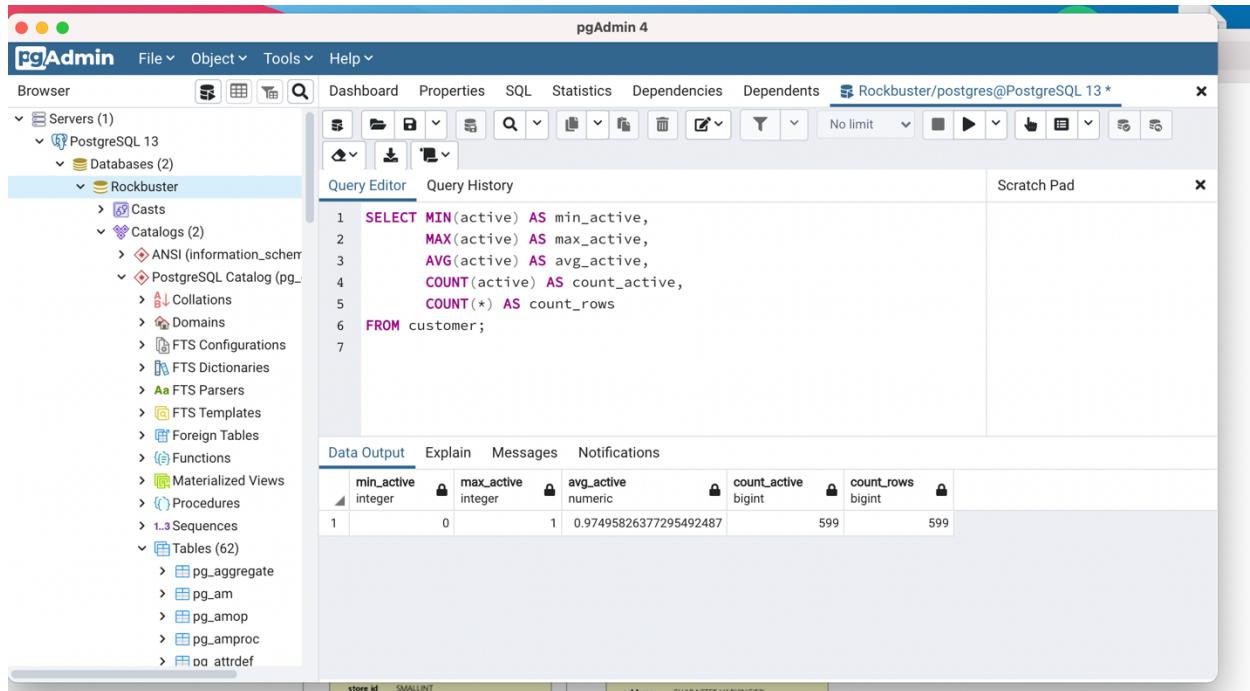
The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (2) > Rockbuster > Tables (62)', the 'film' table is selected. The main area displays a SQL query in the 'Query Editor' tab:

```
1 SELECT MIN(replacement_cost) AS min_replacementcost,
2        MAX(replacement_cost) AS max_replacementcost,
3        AVG(replacement_cost) AS avg_replacementcost,
4        COUNT(replacement_cost) AS count_replacementcost,
5        COUNT(*) AS count_rows
6 FROM film;
7
```

The 'Data Output' tab shows the results of the query:

	min_replacementcost	max_replacementcost	avg_replacementcost	count_replacementcost	count_rows
1	9.99	29.99	19.984000000000000	1000	1000

CUSTOMER TABLE



The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (2) > Rockbuster > Tables (62)', the 'customer' table is selected. The main area displays a SQL query in the 'Query Editor' tab:

```
1 SELECT MIN(active) AS min_active,
2        MAX(active) AS max_active,
3        AVG(active) AS avg_active,
4        COUNT(active) AS count_active,
5        COUNT(*) AS count_rows
6 FROM customer;
7
```

The 'Data Output' tab shows the results of the query:

	min_active	max_active	avg_active	count_active	count_rows
1	0	1	0.97495826377295492487	599	599

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.

Personally, I prefer using SQL because it's easier, as it's more efficient and much faster than Excel. In addition, SQL gives you more options to analyze your data simultaneously.