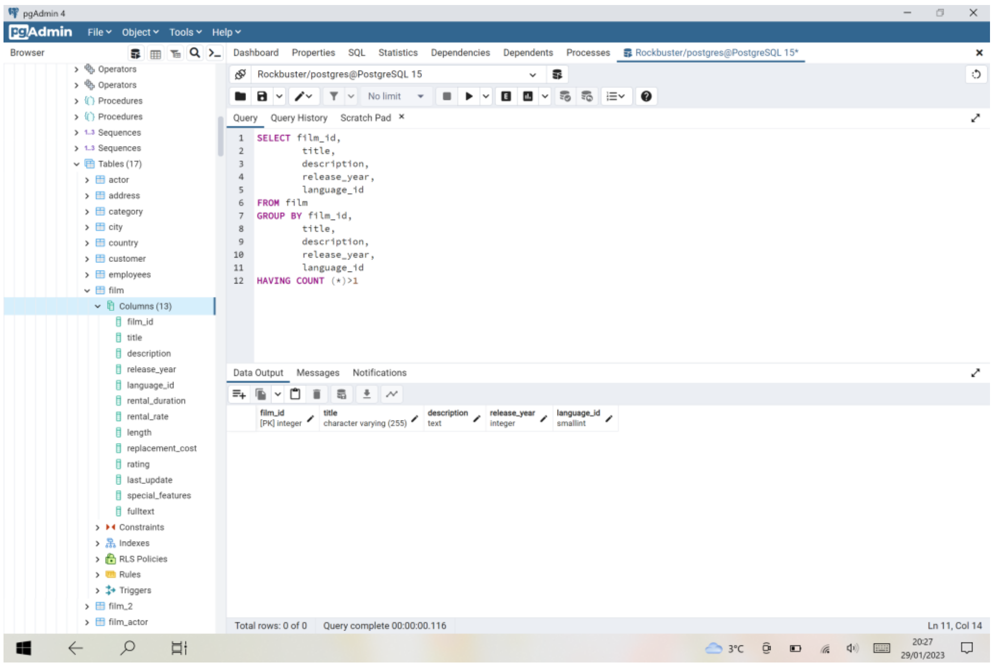
ANSWERS 3.6

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.

   ## FOR FINDING DUPLICATE DATA

   ### FILM TABLE

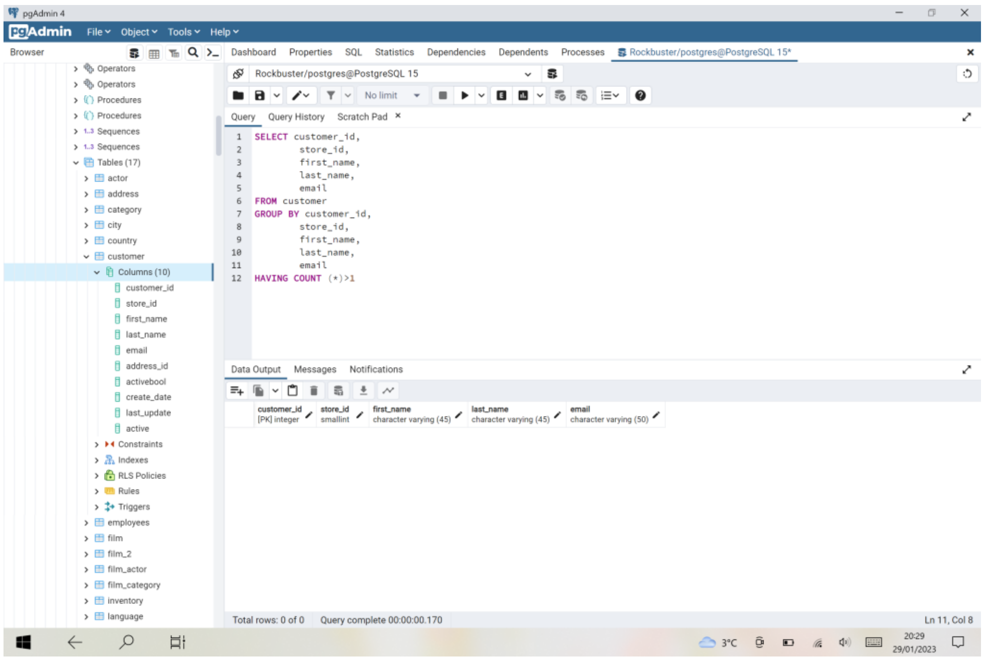


```
SELECT film_id,
       title,
       description,
       release_year,
       language_id
FROM film
GROUP BY film_id,
       title,
       description,
       release_year,
       language_id
HAVING COUNT (*)>1
```

```
film_id        title                        description   release_year   language_id
[PK] integer   character varying (255)      text          integer        smallint
```

Total rows: 0 of 0   Query complete 00:00:00.116   Ln 11, Col 14

CUSTOMER TABLE



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

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

Total rows: 0 of 0   Query complete 00:00:00.170   Ln 11, Col 8

ANSWERS 3.6

Both the film and customer tables have no duplicate records, but it they did we could use 2 methods to clean the data:-

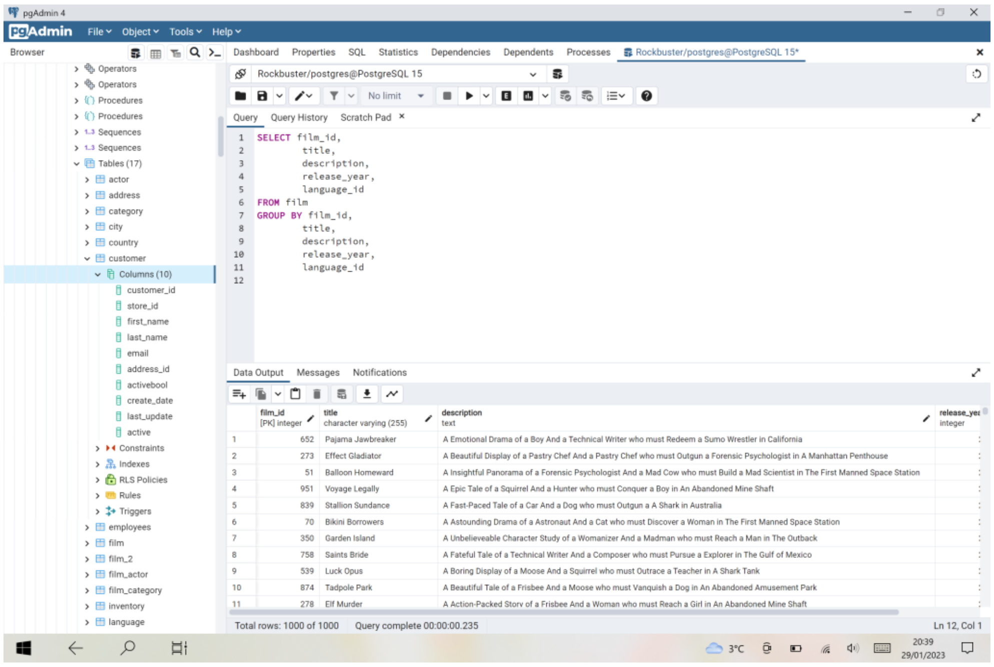
a) We could create a virtual VIEW table, where we could use only unique values

b) We could delete all the duplicate values and then use the existing data.

FOR FINDING NON-UNIFORM DATA

We can either use DISTINCT or GROUP BY to get unique records. Both the table have unique data, but if there was non-uniform data in any column, then we could use UPDATE to get uniform values.

FILM TABLE USING GROUP BY



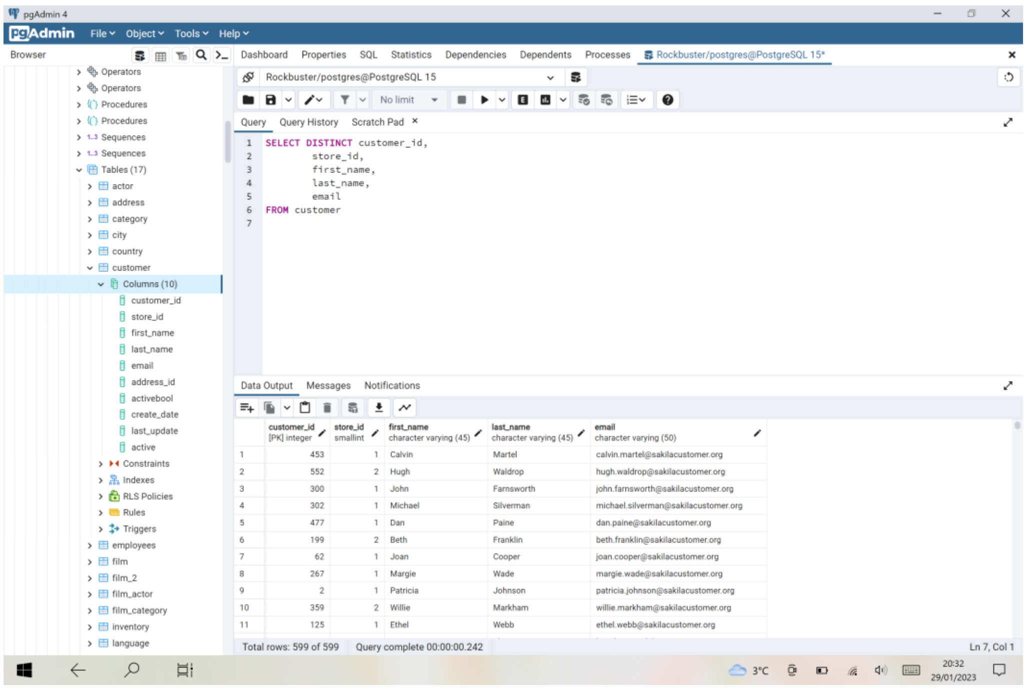
```
SELECT film_id,
        title,
        description,
        release_year,
        language_id
FROM film
GROUP BY film_id,
        title,
        description,
        release_year,
        language_id
```

	film_id [PK] integer	title character varying (255)	description text	release_yea integer
1	652	Pajama Jawbreaker	A Emotional Drama of a Boy And a Technical Writer who must Redeem a Sumo Wrestler in California	
2	273	Effect Gladiator	A Beautiful Display of a Pastry Chef And a Pastry Chef who must Outgun a Forensic Psychologist in A Manhattan Penthouse	
3	51	Balloon Homeward	A Insightful Panorama of a Forensic Psychologist And a Mad Cow who must Build a Mad Scientist in The First Manned Space Station	
4	951	Voyage Legally	A Epic Tale of a Squirrel And a Hunter who must Conquer a Boy in An Abandoned Mine Shaft	
5	839	Stallion Sundance	A Fast-Paced Tale of a Car And a Dog who must Outgun a A Shark in Australia	
6	70	Bikini Borrowers	A Astounding Drama of a Astronaut And a Cat who must Discover a Woman in The First Manned Space Station	
7	350	Garden Island	A Unbelieveable Character Study of a Womanizer And a Madman who must Reach a Man in The Outback	
8	758	Saints Bride	A Fateful Tale of a Technical Writer And a Composer who must Pursue a Explorer in The Gulf of Mexico	
9	539	Luck Opus	A Boring Display of a Moose And a Squirrel who must Outrace a Teacher in A Shark Tank	
10	874	Tadpole Park	A Beautiful Tale of a Frisbee And a Moose who must Vanquish a Dog in An Abandoned Amusement Park	
11	278	Elf Murder	A Action-Packed Story of a Frisbee And a Woman who must Reach a Girl in An Abandoned Mine Shaft	

Total rows: 1000 of 1000 Query complete 00:00:00.235 Ln 12, Col 1

ANSWERS 3.6

CUSTOMER TABLE using DISTINCT



```
SELECT DISTINCT customer_id,
        store_id,
        first_name,
        last_name,
        email
FROM customer
```

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)
1	453	1	Calvin	Martel	calvin.martel@sakilacustomer.org
2	552	2	Hugh	Waldrop	hugh.waldrop@sakilacustomer.org
3	300	1	John	Farnsworth	john.farnsworth@sakilacustomer.org
4	302	1	Michael	Silverman	michael.silverman@sakilacustomer.org
5	477	1	Dan	Paine	dan.paine@sakilacustomer.org
6	199	2	Beth	Franklin	beth.franklin@sakilacustomer.org
7	62	1	Joan	Cooper	joan.cooper@sakilacustomer.org
8	267	1	Margie	Wade	margie.wade@sakilacustomer.org
9	2	1	Patricia	Johnson	patricia.johnson@sakilacustomer.org
10	359	2	Willie	Markham	willie.markham@sakilacustomer.org
11	125	1	Ethel	Webb	ethel.webb@sakilacustomer.org

```
Total rows: 599 of 599    Query complete 00:00:00.242
```

FOR FINDING MISSING DATA

This data has no missing data. Missing values are often null, empty, or replaced with a dummy or default value instead. Like incorrect data, there's no quick fix for missing data. If there is a lot of missing data then we can either ignore the data completely or we can impute average values .

-Ignore columns with high percentage of missing values

SELECT column1, column2, column4

--column3 ignored in select because it has a lot of missing values

FROM tablename

--Imputing missing values with the average value

UPDATE tablename

SET=AVG (column1)

WHERE column1 IS NULL

ANSWERS 3.6

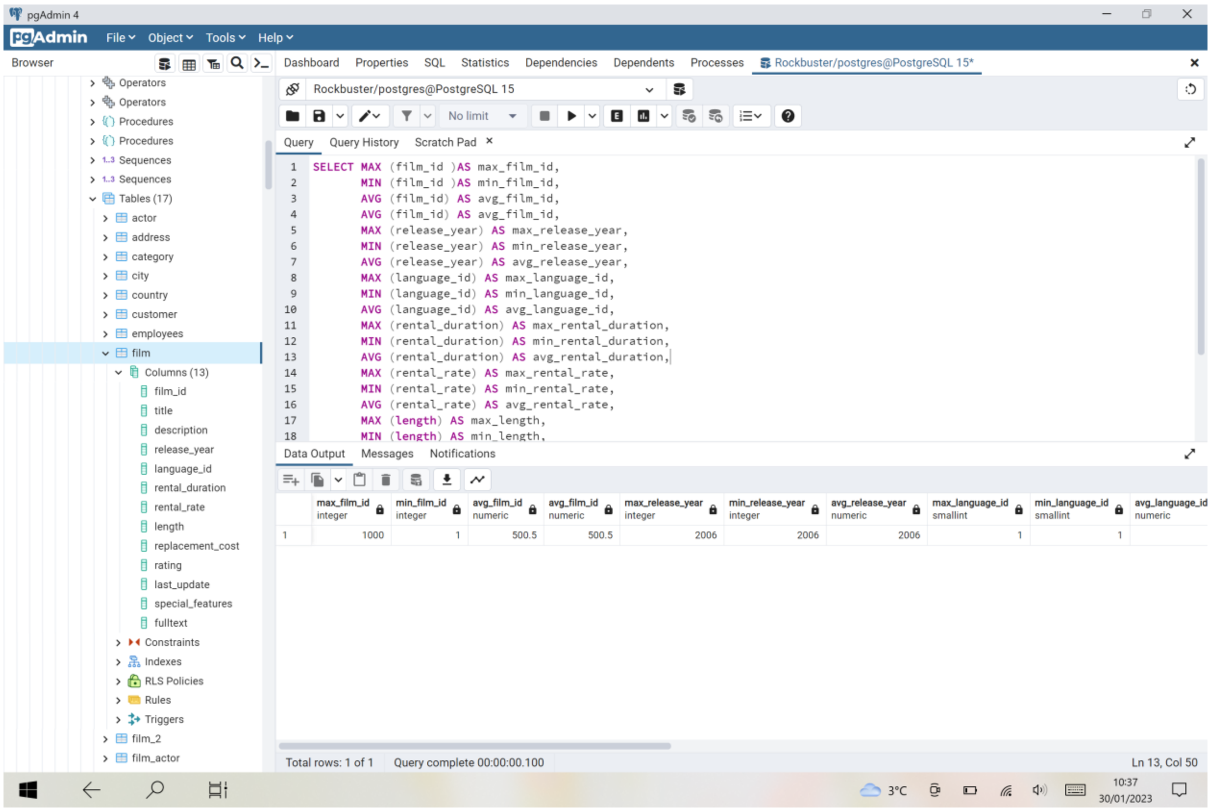
2. **Summarize your data:** Use SQL to calculate descriptive statistics for both the film table and the customer table. For numerical columns, this means finding the minimum, maximum, and average values. For non-numerical columns, calculate the mode value.

FILM TABLE (NUMERICAL VALUES)

```
SELECT MAX (film_id )AS max_film_id,
        MIN (film_id )AS min_film_id,
        AVG (film_id) AS avg_film_id,
        AVG (film_id) AS avg_film_id,
        MAX (release_year) AS max_release_year,
        MIN (release_year) AS min_release_year,
        AVG (release_year) AS avg_release_year,
        MAX (language_id) AS max_language_id,
        MIN (language_id) AS min_language_id,
        AVG (language_id) AS avg_language_id,
        MAX (rental_duration) AS max_rental_duration,
        MIN (rental_duration) AS min_rental_duration,
        AVG (rental_duration) AS avg_rental_duration,
        MAX (rental_rate) AS max_rental_rate,
        MIN (rental_rate) AS min_rental_rate,
        AVG (rental_rate) AS avg_rental_rate,
        MAX (length) AS max_length,
        MIN (length) AS min_length,
        AVG (length) AS avg_length,
        MAX (replacement_cost) AS max_replacement_cost,
        MIN (replacement_cost) AS min_replacement_cost,
        AVG (replacement_cost) AS avg_replacement_cost

FROM film
```

ANSWERS 3.6



```
SELECT MAX (film_id )AS max_film_id,
       MIN (film_id )AS min_film_id,
       AVG (film_id) AS avg_film_id,
       AVG (film_id) AS avg_film_id,
       MAX (release_year) AS max_release_year,
       MIN (release_year) AS min_release_year,
       AVG (release_year) AS avg_release_year,
       MAX (language_id) AS max_language_id,
       MIN (language_id) AS min_language_id,
       AVG (language_id) AS avg_language_id,
       MAX (rental_duration) AS max_rental_duration,
       MIN (rental_duration) AS min_rental_duration,
       AVG (rental_duration) AS avg_rental_duration,
       MAX (rental_rate) AS max_rental_rate,
       MIN (rental_rate) AS min_rental_rate,
       AVG (rental_rate) AS avg_rental_rate,
       MAX (length) AS max_length,
       MIN (length) AS min_length,
```

	max_film_id integer	min_film_id integer	avg_film_id numeric	avg_film_id numeric	max_release_year integer	min_release_year integer	avg_release_year numeric	max_language_id smallint	min_language_id smallint	avg_language_id numeric
1	1000	1	500.5	500.5	2006	2006	2006	1	1	

Total rows: 1 of 1   Query complete 00:00:00.100   Ln 13, Col 50

FILM TABLE (NON-NUMERICAL VALUES)

SELECT mode() WITHIN GROUP (ORDER BY title)AS mode_title,

WITHIN GROUP (ORDER BY description)AS mode_description,

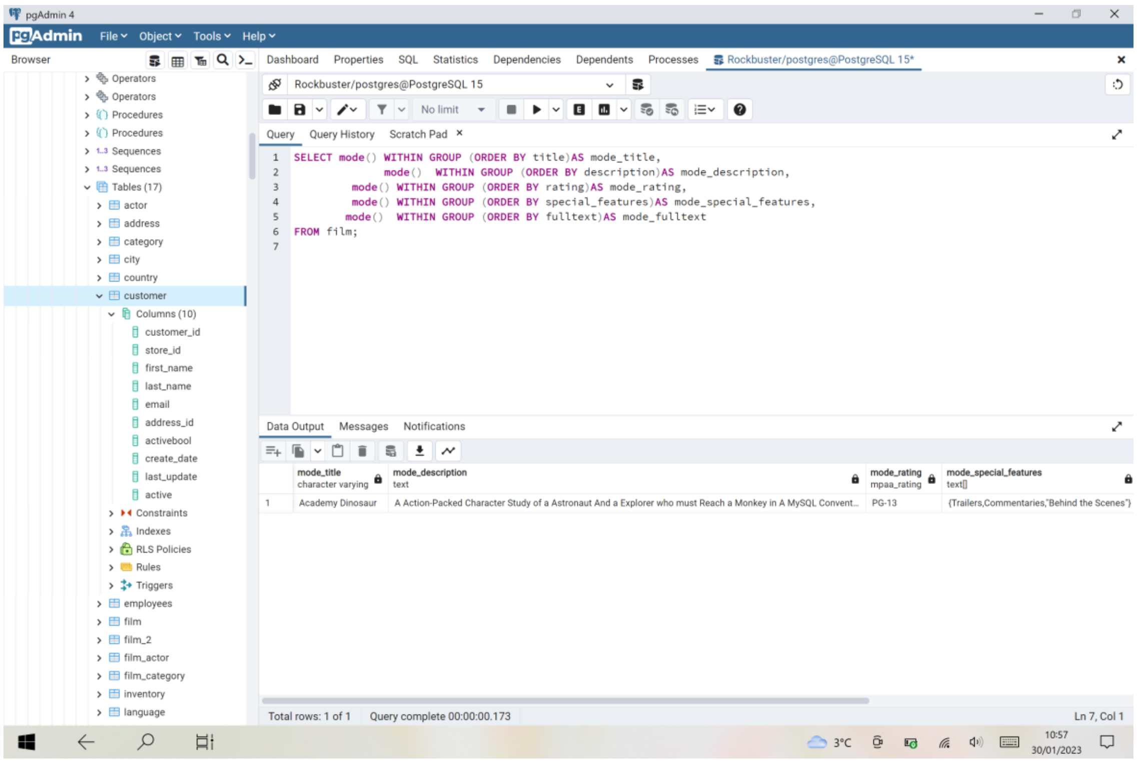
WITHIN GROUP (ORDER BY rating)AS mode_rating,

WITHIN GROUP (ORDER BY special_features)AS mode_special_features,

WITHIN GROUP (ORDER BY fulltext)AS mode_fulltext

FROM film;

ANSWERS 3.6

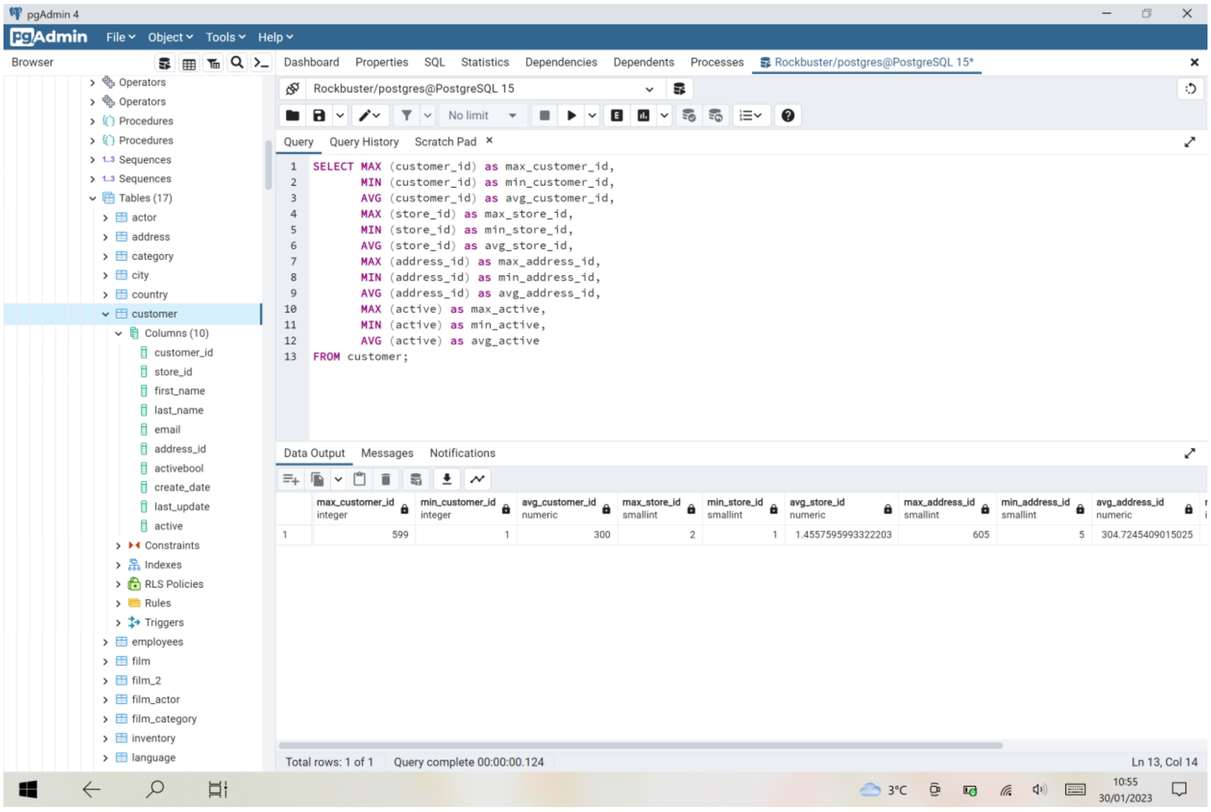


```
SELECT mode() WITHIN GROUP (ORDER BY title)AS mode_title,
            mode()  WITHIN GROUP (ORDER BY description)AS mode_description,
        mode() WITHIN GROUP (ORDER BY rating)AS mode_rating,
        mode() WITHIN GROUP (ORDER BY special_features)AS mode_special_features,
       mode()  WITHIN GROUP (ORDER BY fulltext)AS mode_fulltext
FROM film;
```

	mode_title character varying	mode_description text	mode_rating mpaa_rating	mode_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"}

Total rows: 1 of 1   Query complete 00:00:00.173   Ln 7, Col 1

CUSTOMER TABLE (NUMERICAL VALUES)



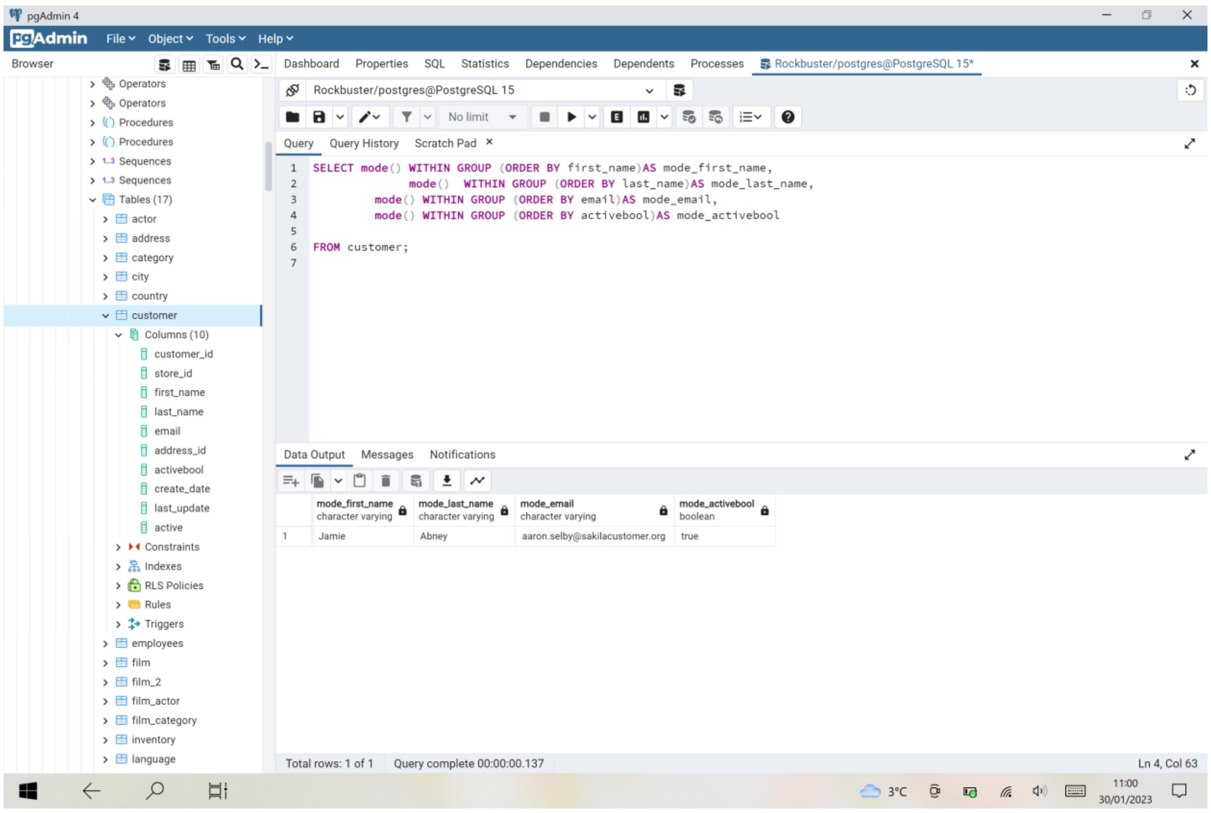
```
SELECT MAX (customer_id) as max_customer_id,
       MIN (customer_id) as min_customer_id,
       AVG (customer_id) as avg_customer_id,
       MAX (store_id) as max_store_id,
       MIN (store_id) as min_store_id,
       AVG (store_id) as avg_store_id,
       MAX (address_id) as max_address_id,
       MIN (address_id) as min_address_id,
       AVG (address_id) as avg_address_id,
       MAX (active) as max_active,
       MIN (active) as min_active,
       AVG (active) as avg_active
FROM customer;
```

	max_customer_id integer	min_customer_id integer	avg_customer_id numeric	max_store_id smallint	min_store_id smallint	avg_store_id numeric	max_address_id smallint	min_address_id smallint	avg_address_id numeric
1	599	1	300	2	1	1.4557595993322203	605	5	304.7245409015025

Total rows: 1 of 1   Query complete 00:00:00.124   Ln 13, Col 14

ANSWERS 3.6

CUSTOMER TABLE (NON-NUMERICAL VALUES)



```
SELECT mode() WITHIN GROUP (ORDER BY first_name)AS mode_first_name,
              mode()  WITHIN GROUP (ORDER BY last_name)AS mode_last_name,
         mode() WITHIN GROUP (ORDER BY email)AS mode_email,
         mode() WITHIN GROUP (ORDER BY activebool)AS mode_activebool

FROM customer;
```

	mode_first_name character varying	mode_last_name character varying	mode_email character varying	mode_activebool boolean
1	Jamie	Abney	aaron.selby@sakilacustomer.org	true

Total rows: 1 of 1   Query complete 00:00:00.137

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.

    - In my view, I feel SQL is a lot better as a tool for data profiling. It is irrespective of the data size; we get quicker results without scanning through the complete data.
    - We can just write the relevant queries and find out the dirty data and clean it.
    - With appropriate queries we can find out the unique data.