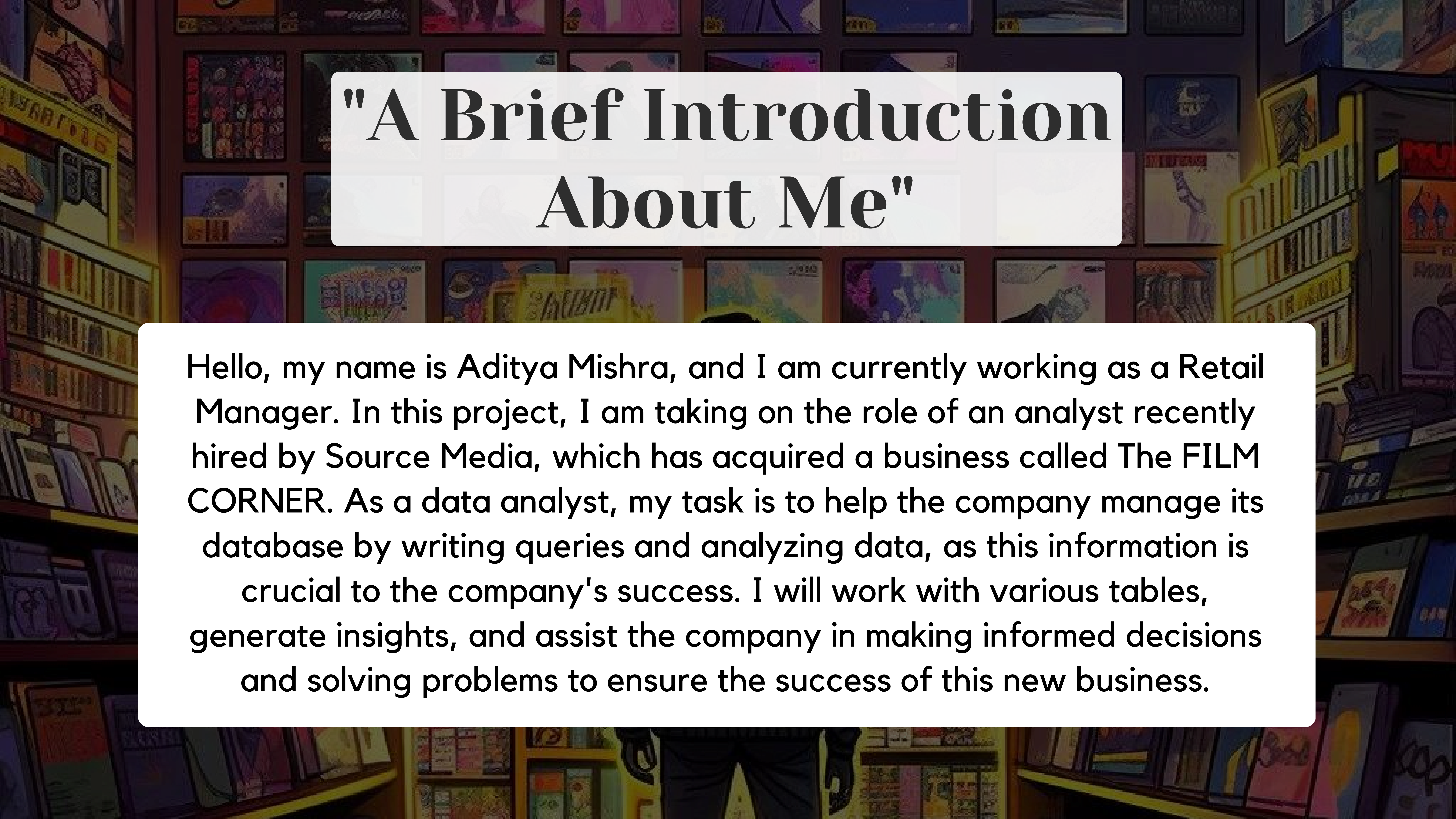
THE-FILM-CORNER

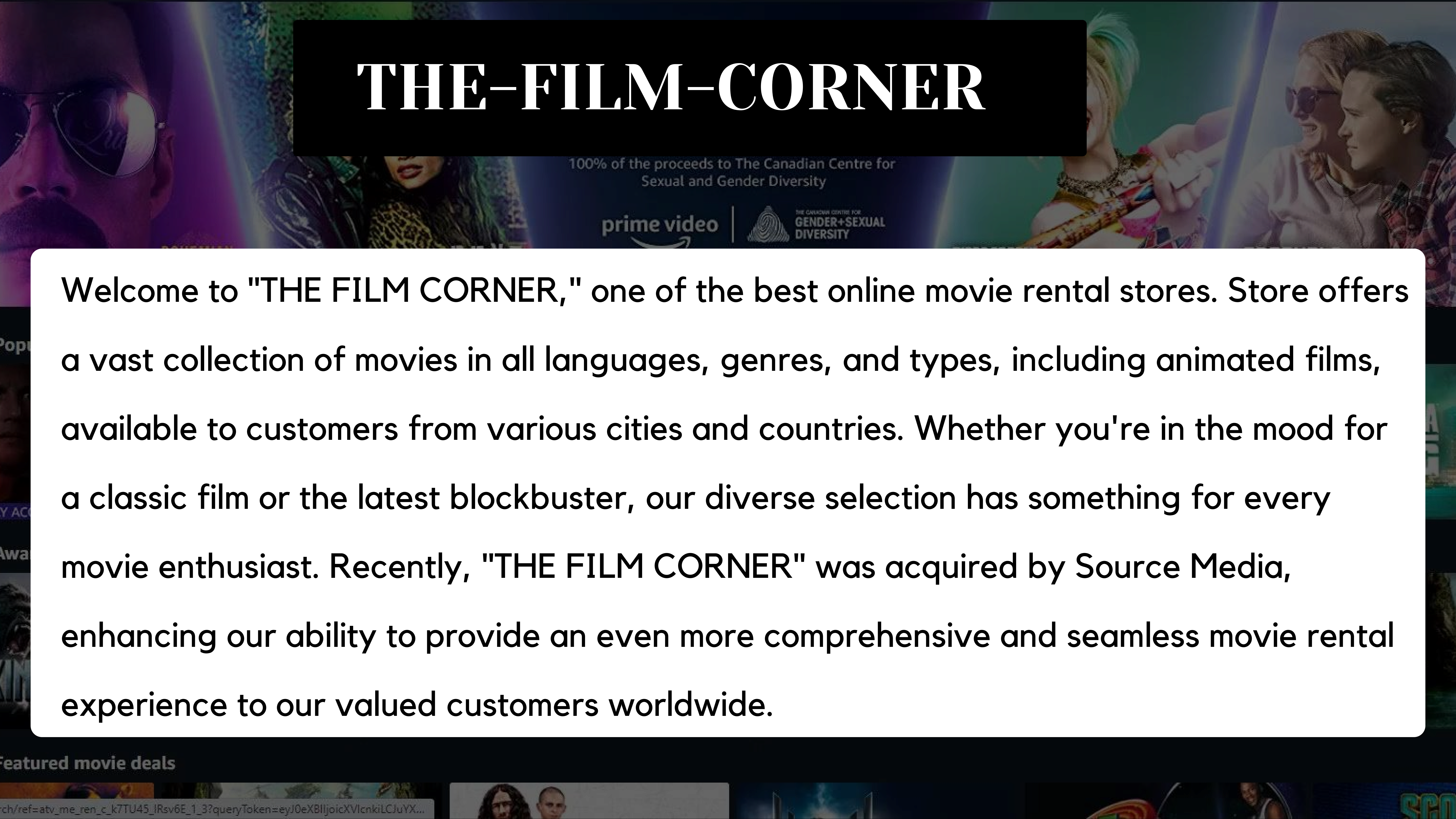
Online movie rental store

Revenue insights
Data analysis
Data Querying
Sorting and Filtering



"A Brief Introduction About Me"

Hello, my name is Aditya Mishra, and I am currently working as a Retail Manager. In this project, I am taking on the role of an analyst recently hired by Source Media, which has acquired a business called The FILM CORNER. As a data analyst, my task is to help the company manage its database by writing queries and analyzing data, as this information is crucial to the company's success. I will work with various tables, generate insights, and assist the company in making informed decisions and solving problems to ensure the success of this new business.

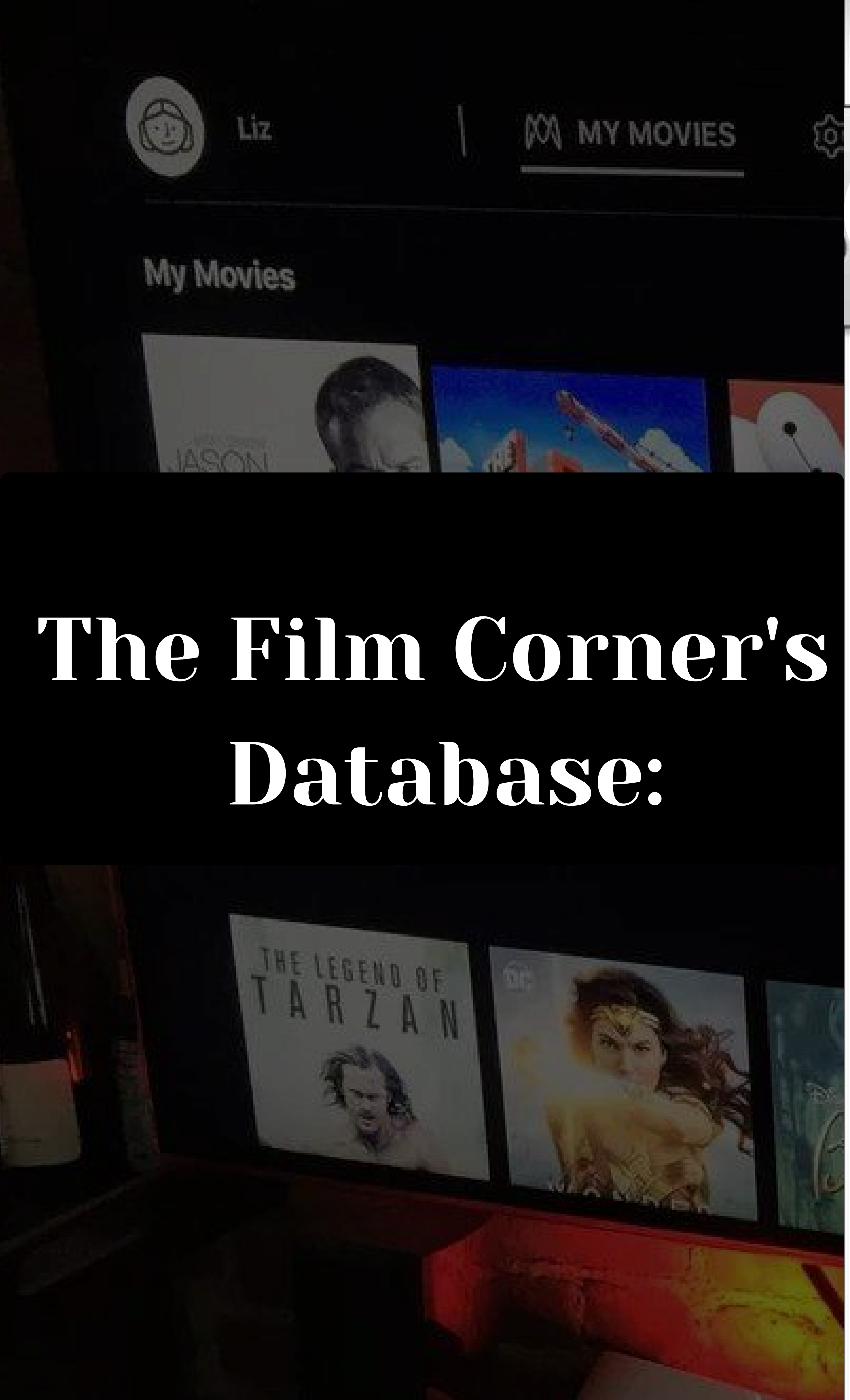

100% of the proceeds to The Canadian Centre for Sexual and Gender Diversity
prime video
THE CANADIAN CENTRE FOR GENDER+SEXUAL DIVERSITY
Featured movie deals
ch/ref=atv_me_ren_c_k7TU45_IRsv6E_1_3?queryToken=eyJ0eXBlIjoicXVlcnkiLCJuYX...

THE-FILM-CORNER

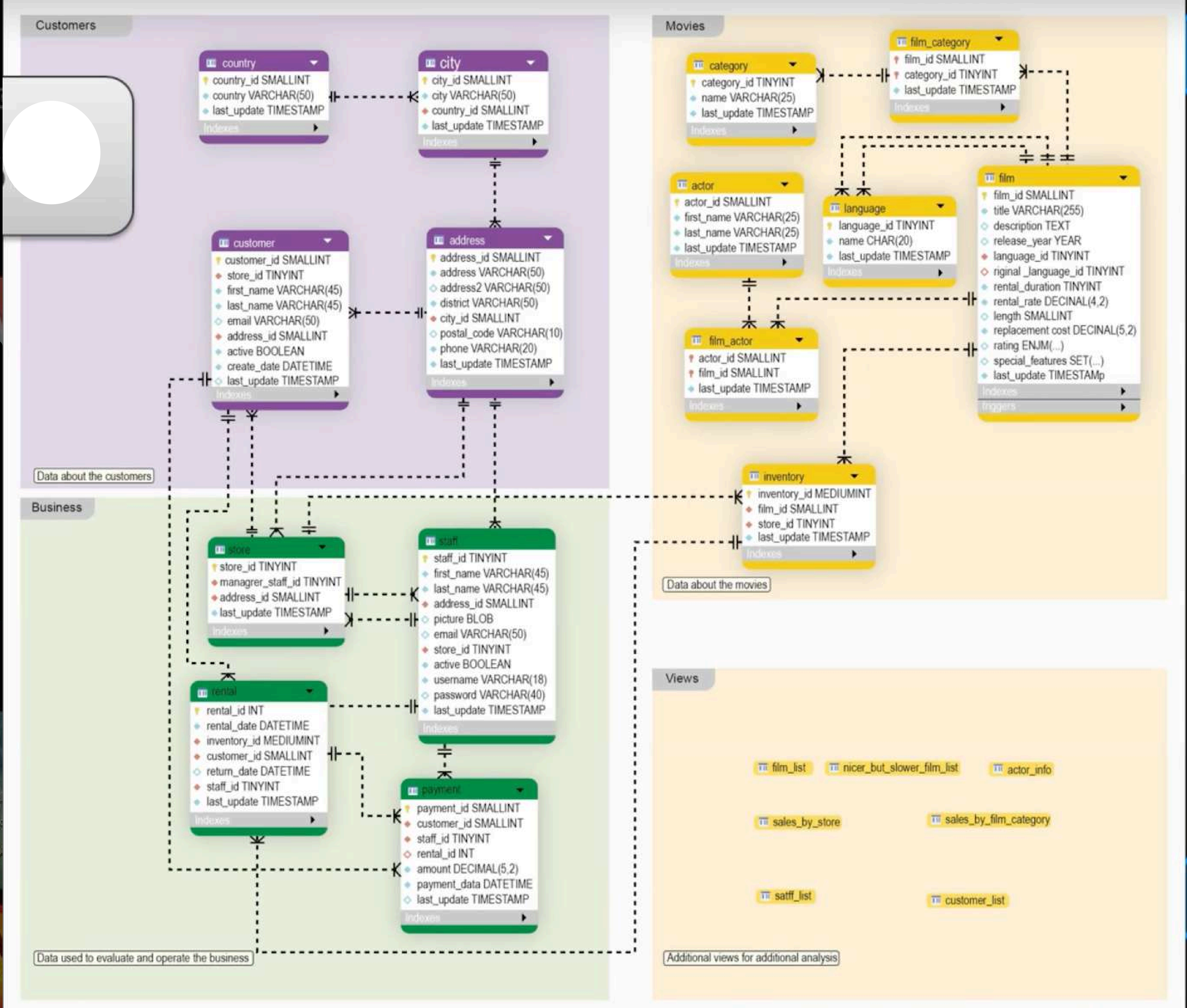
Welcome to "THE FILM CORNER," one of the best online movie rental stores. Store offers a vast collection of movies in all languages, genres, and types, including animated films, available to customers from various cities and countries. Whether you're in the mood for a classic film or the latest blockbuster, our diverse selection has something for every movie enthusiast. Recently, "THE FILM CORNER" was acquired by Source Media, enhancing our ability to provide an even more comprehensive and seamless movie rental experience to our valued customers worldwide.

NDANGONOW
MOVIES
TV SHOWS
GET STARTED
REDEEM
GIFT CARDS
JOIN FANDANGO VIP
THE PHOTOGRAPH
BAD BOYS FOR LIFE
SONIC THE HEDGEHOG
THE ASSISTANT
THE INVISIBLE MAN
LIKE A BOSS
HOME PREMIERE
MOVIES DIRECT FROM THEATERS
THE HUNT
P AND ENJOY
uainted with some of your favorite films
INDIANA JONES and the RAIDERS of the LOST ARK
GONE WITH THE WIND
A Few Good Men

Let's dive in !!



The Film Corner's Database:



Customers

country
- country_id SMALLINT
- country VARCHAR(50)
- last_update TIMESTAMP
- Indexes

city
- city_id SMALLINT
- city VARCHAR(50)
- country_id SMALLINT
- last_update TIMESTAMP
- Indexes

customer
- customer_id SMALLINT
- store_id TINYINT
- first_name VARCHAR(45)
- last_name VARCHAR(45)
- email VARCHAR(50)
- address_id SMALLINT
- active BOOLEAN
- create_date DATETIME
- last_update TIMESTAMP
- Indexes

address
- address_id SMALLINT
- address VARCHAR(50)
- address2 VARCHAR(50)
- district VARCHAR(50)
- city_id SMALLINT
- postal_code VARCHAR(10)
- phone VARCHAR(20)
- last_update TIMESTAMP
- Indexes

Data about the customers

Movies

category
- category_id TINYINT
- name VARCHAR(25)
- last_update TIMESTAMP
- Indexes

film_category
- film_id SMALLINT
- category_id TINYINT
- last_update TIMESTAMP
- Indexes

actor
- actor_id SMALLINT
- first_name VARCHAR(25)
- last_name VARCHAR(25)
- last_update TIMESTAMP
- Indexes

language
- language_id TINYINT
- name CHAR(20)
- last_update TIMESTAMP
- Indexes

film
- film_id SMALLINT
- title VARCHAR(255)
- description TEXT
- release_year YEAR
- language_id TINYINT
- riginal _language_id TINYINT
- rental_duration TINYINT
- rental_rate DECINAL(4,2)
- length SMALLINT
- replacement cost DECINAL(5,2)
- rating ENJM(...)
- special_features SET(...)
- last_update TIMESTAMp
- Indexes
- triggers

film_actor
- actor_id SMALLINT
- film_id SMALLINT
- last_update TIMESTAMP
- Indexes

inventory
- inventory_id MEDIUMINT
- film_id SMALLINT
- store_id TINYINT
- last_update TIMESTAMP
- Indexes

Data about the movies

Business

store
- store_id TINYINT
- managrer_staff_id TINYINT
- address_id SMALLINT
- last_update TIMESTAMP
- Indexes

staff
- staff_id TINYINT
- first_name VARCHAR(45)
- last_name VARCHAR(45)
- address_id SMALLINT
- picture BLOB
- email VARCHAR(50)
- store_id TINYINT
- active BOOLEAN
- username VARCHAR(18)
- password VARCHAR(40)
- last_update TIMESTAMP
- Indexes

rental
- rental_id INT
- rental_date DATETIME
- inventory_id MEDIUMINT
- customer_id SMALLINT
- return_date DATETIME
- staff_id TINYINT
- last_update TIMESTAMP
- Indexes

payment
- payment_id SMALLINT
- customer_id SMALLINT
- staff_id TINYINT
- rental_id INT
- amount DECIMAL(5,2)
- payment_data DATETIME
- last_update TIMESTAMP
- Indexes

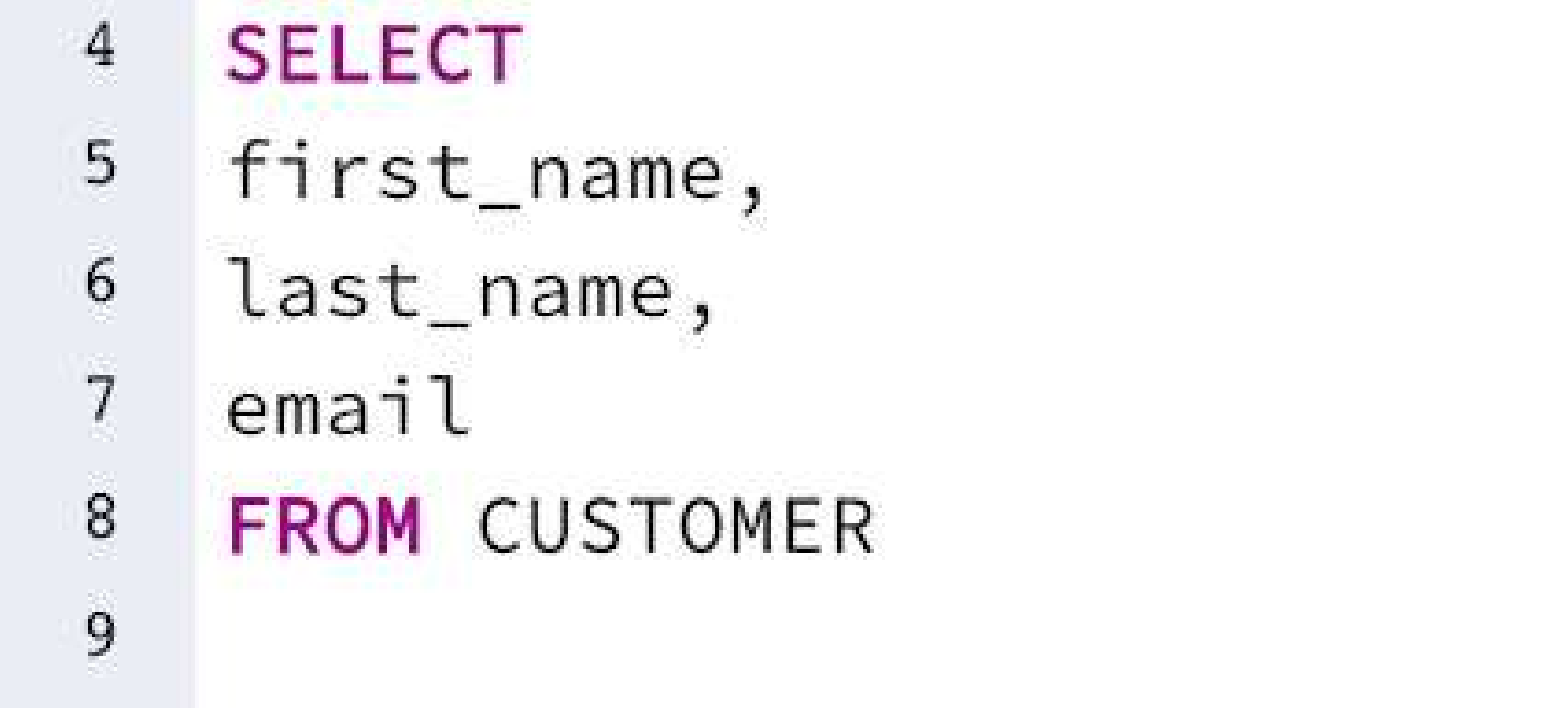
Data used to evaluate and operate the business

Views

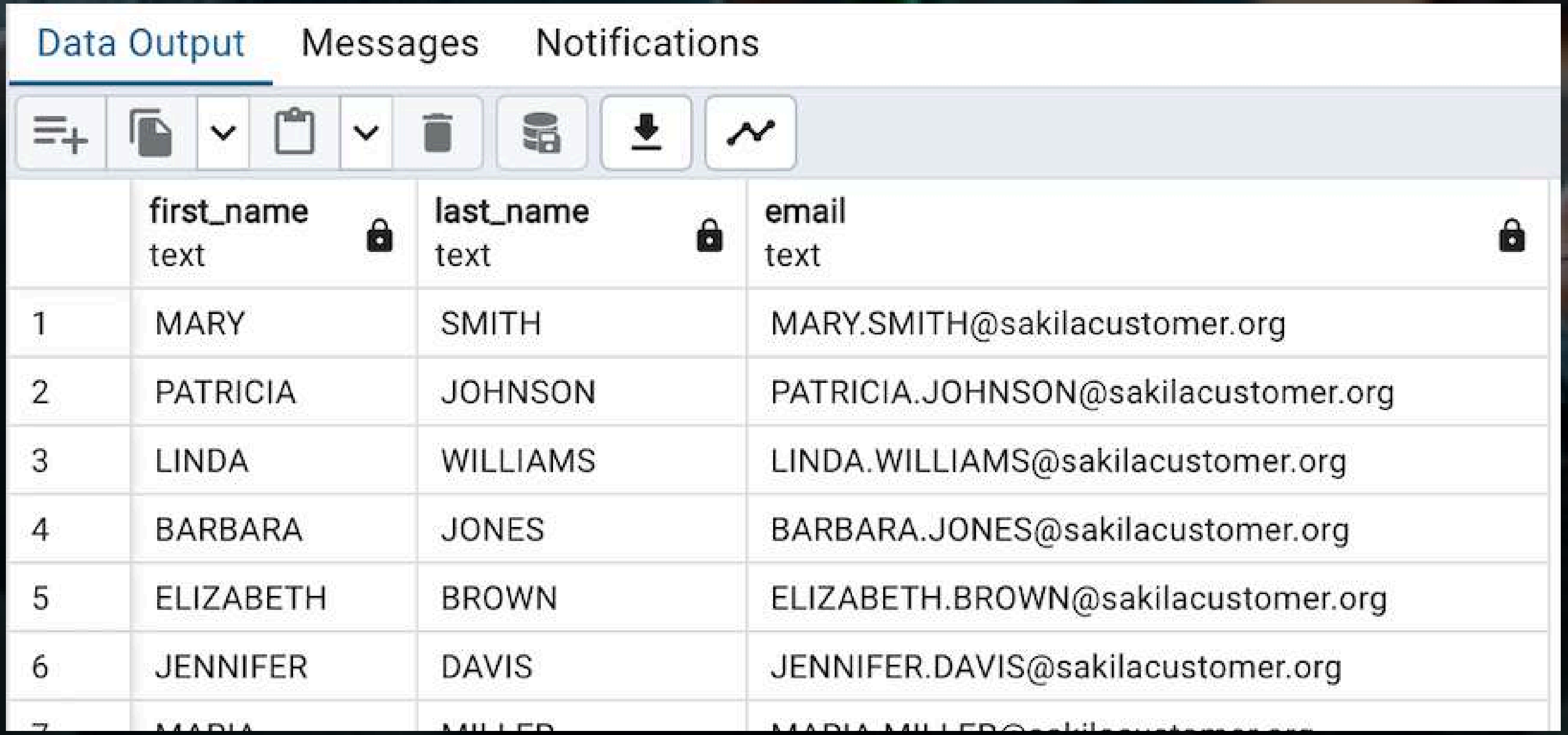
- film_list
- nicer_but_slower_film_list
- actor_info
- sales_by_store
- sales_by_film_category
- satff_list
- customer_list

Additional views for additional analysis

The marketing manager has requested a list of all customers of the rental shop, including each customer's first name,
last name, and email address.

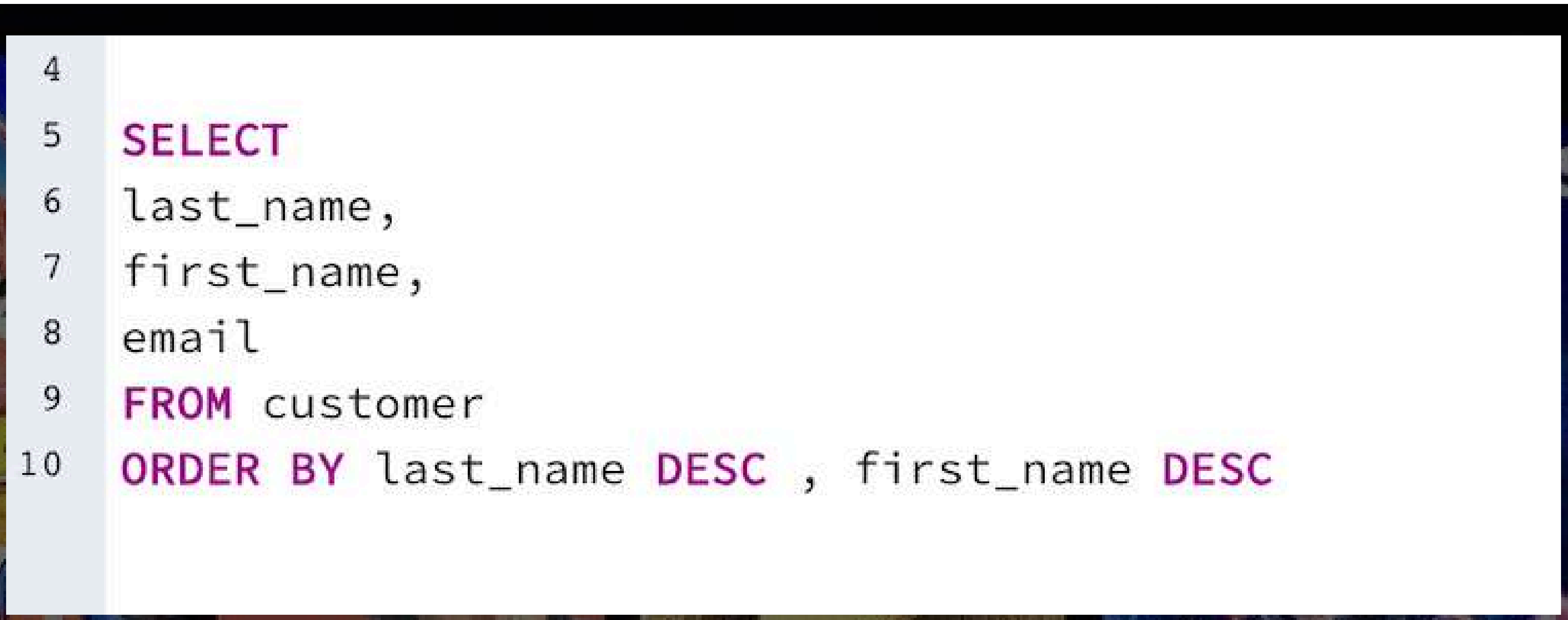


```sql
SELECT
first_name,
last_name,
email
FROM CUSTOMER

```

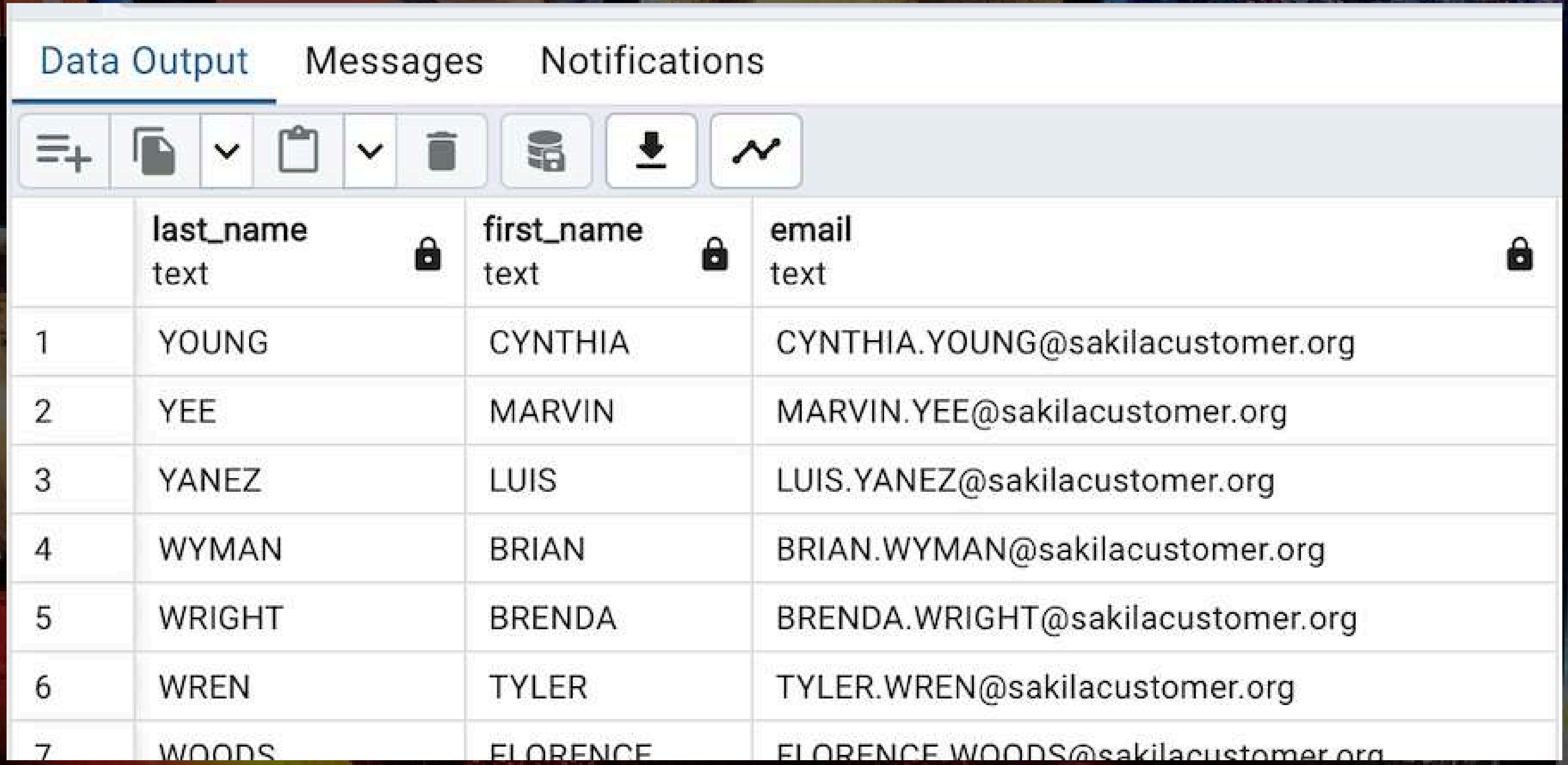


	first_name text	last_name text	email text
1	MARY	SMITH	MARY.SMITH@sakilacustomer.org
2	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org
3	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org
4	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org
5	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org
6	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org
7	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org

The marketing manager now requests that you organize the customer list you previously provided by last name, starting from Z to A. Additionally, in the event that there are multiple entries with the same last name, please use the first name as a secondary criterion for ordering, also from Z to A.

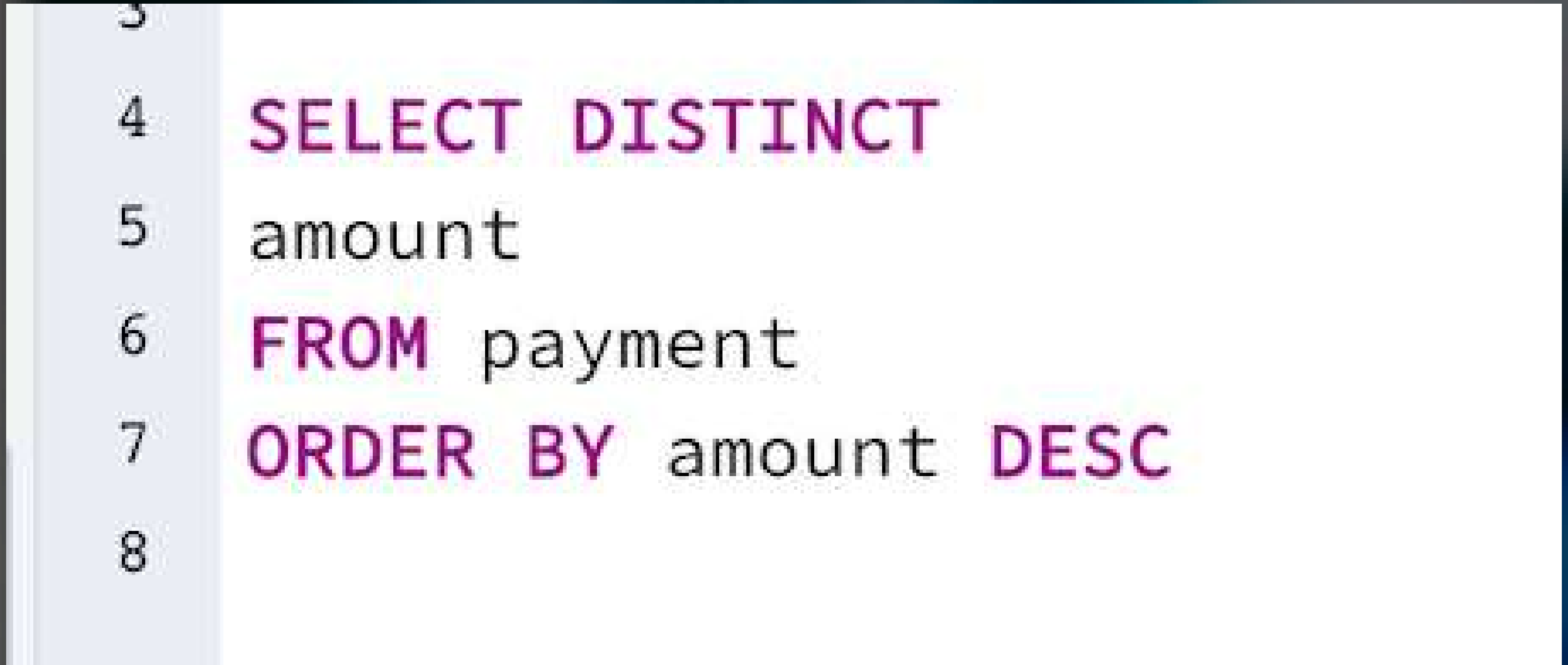


```sql
SELECT
last_name,
first_name,
email
FROM customer
ORDER BY last_name DESC , first_name DESC
```

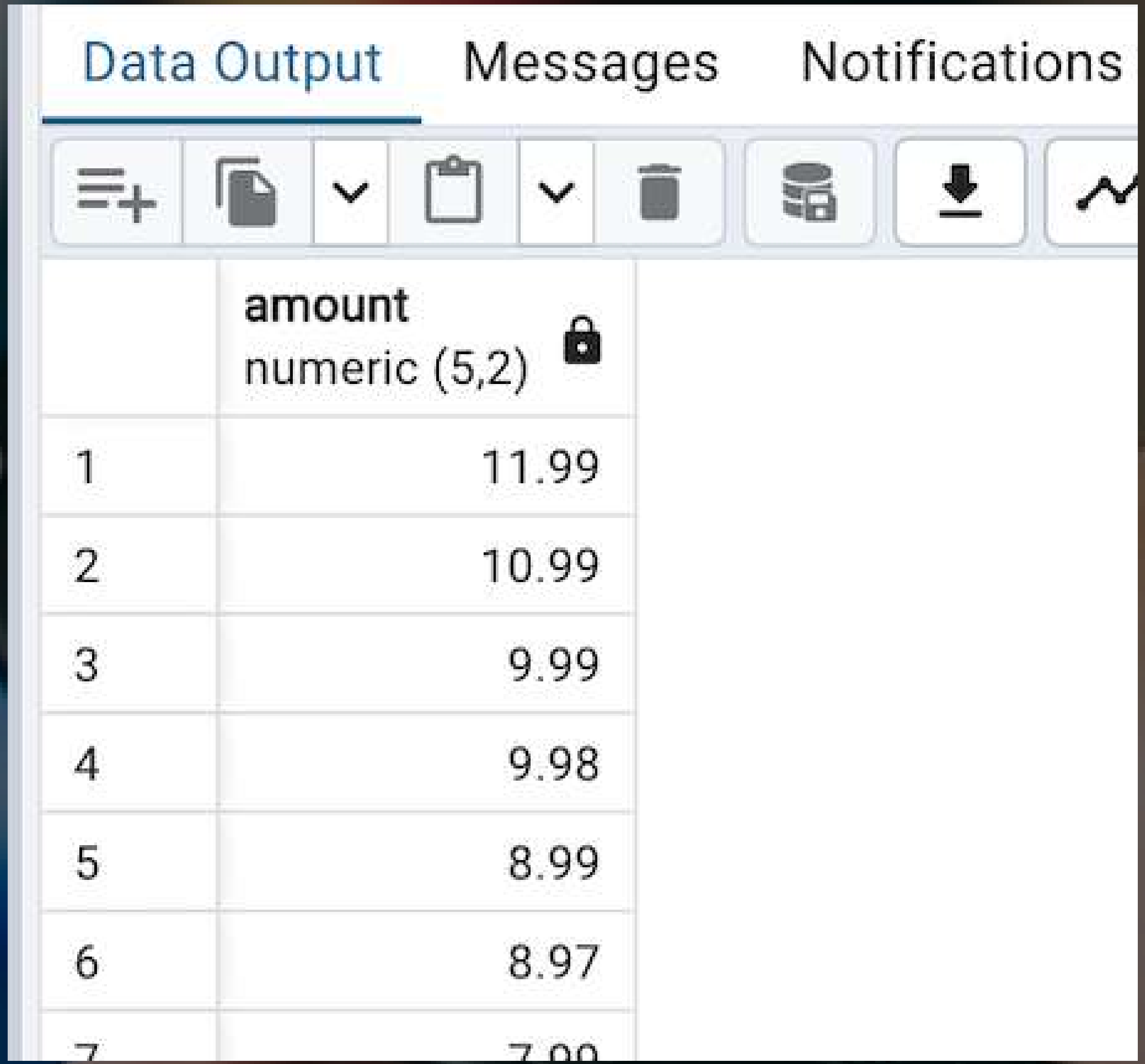


	last_name text	first_name text	email text
1	YOUNG	CYNTHIA	CYNTHIA.YOUNG@sakilacustomer.org
2	YEE	MARVIN	MARVIN.YEE@sakilacustomer.org
3	YANEZ	LUIS	LUIS.YANEZ@sakilacustomer.org
4	WYMAN	BRIAN	BRIAN.WYMAN@sakilacustomer.org
5	WRIGHT	BRENDA	BRENDA.WRIGHT@sakilacustomer.org
6	WREN	TYLER	TYLER.WREN@sakilacustomer.org
7	WOODS	FLORENCE	FLORENCE.WOODS@sakilacustomer.org

A member of the marketing team is inquiring about the various prices that have been paid in the business in the past. To simplify their task, you can also arrange those prices from highest to lowest.

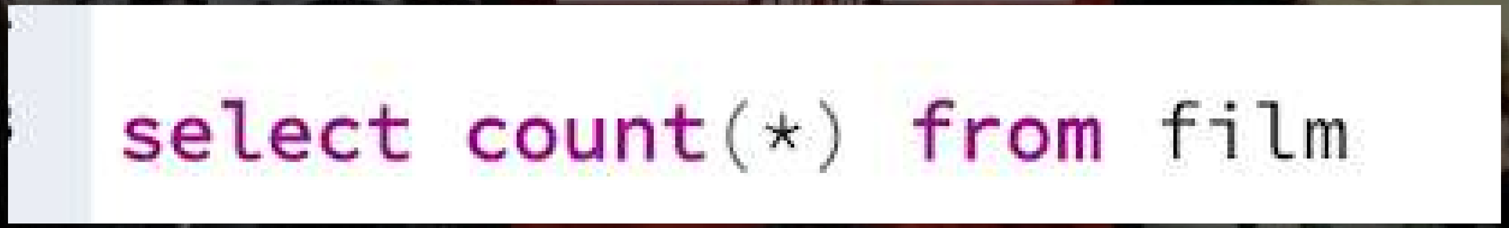


```sql
SELECT DISTINCT
amount
FROM payment
ORDER BY amount DESC
```

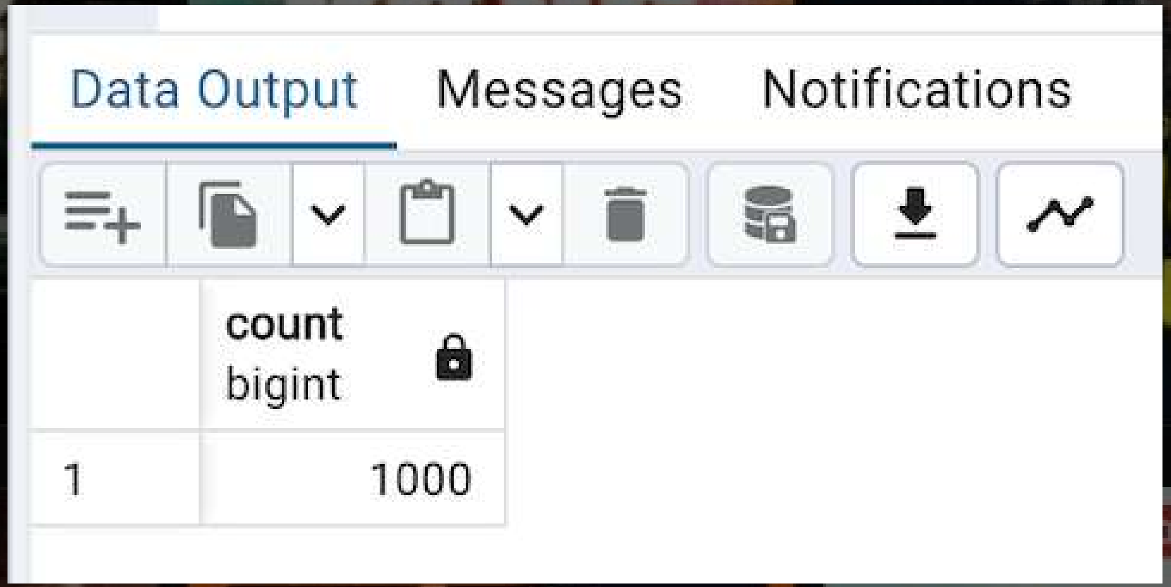


```
   amount
   numeric (5,2)
1          11.99
2          10.99
3           9.99
4           9.98
5           8.99
6           8.97
7           7.99
```

How many films does the company have ?

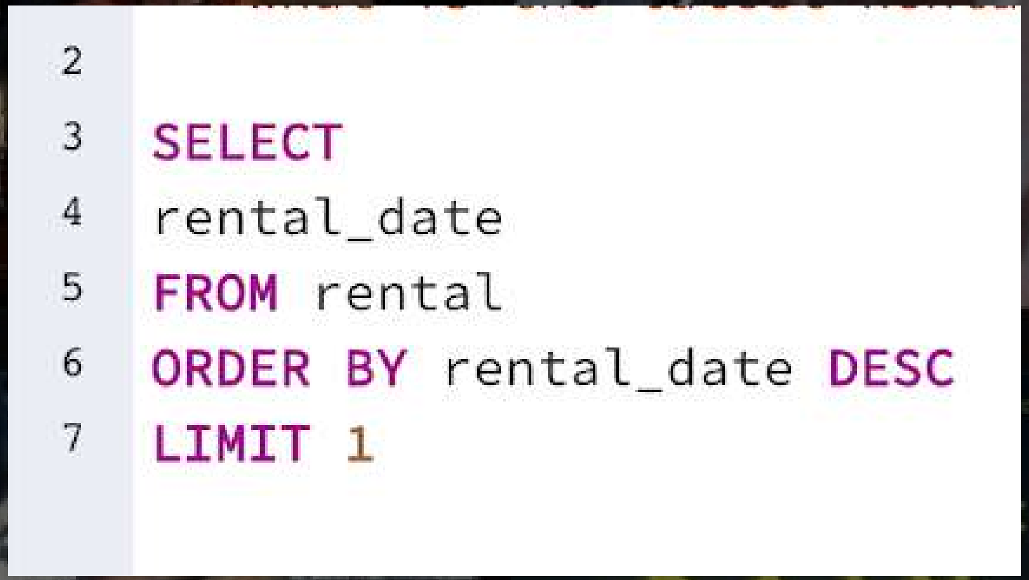


```sql
select count(*) from film
```

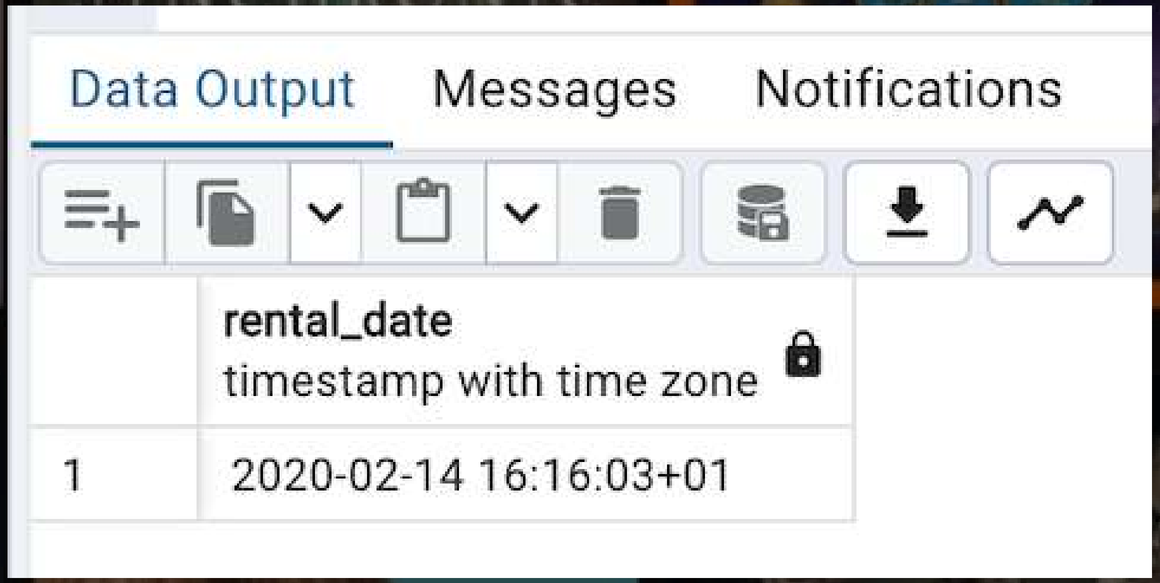


	count bigint
1	1000

What is the latest Rental date ?

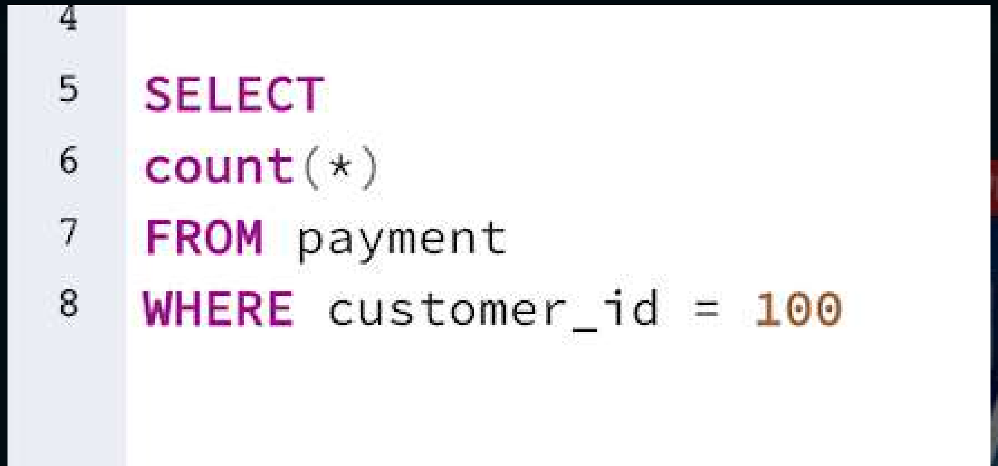


```sql
SELECT
rental_date
FROM rental
ORDER BY rental_date DESC
LIMIT 1
```

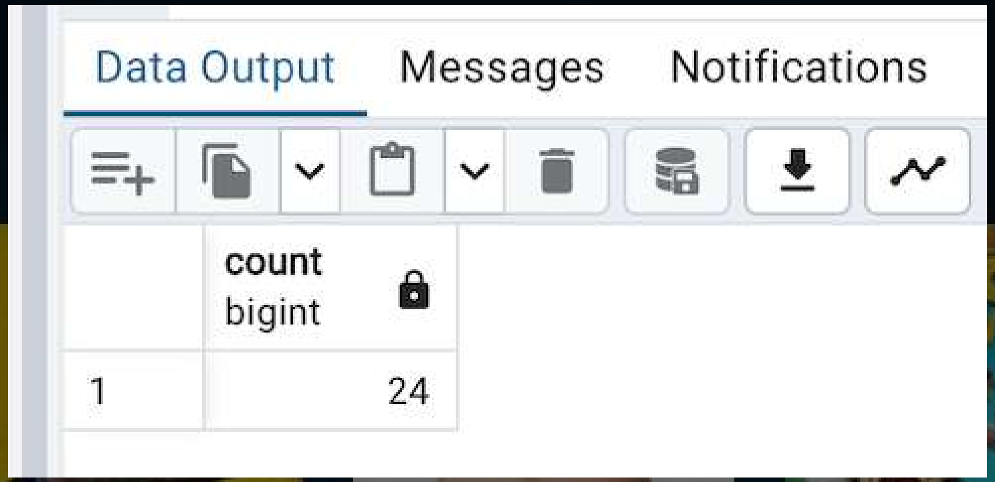


	rental_date timestamp with time zone
1	2020-02-14 16:16:03+01

The marketing team is organizing an event during which they intend to provide a gift voucher to the customer identified by customer ID 100. Initially, they need to determine the total number of payments made by this customer, as they plan to issue a $1 voucher per payment made.

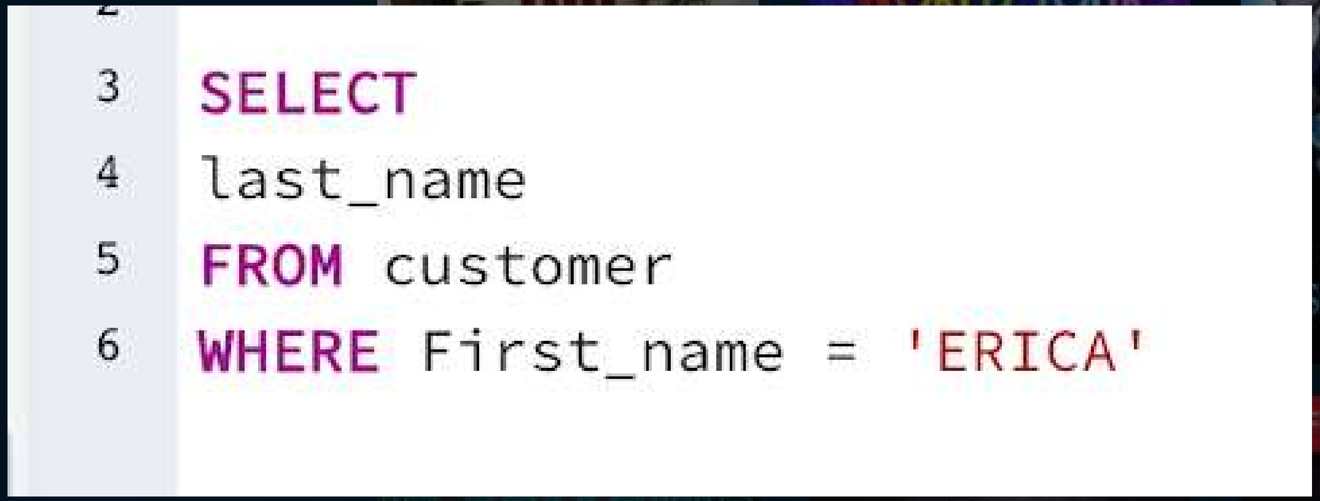


```
SELECT
count(*)
FROM payment
WHERE customer_id = 100
```

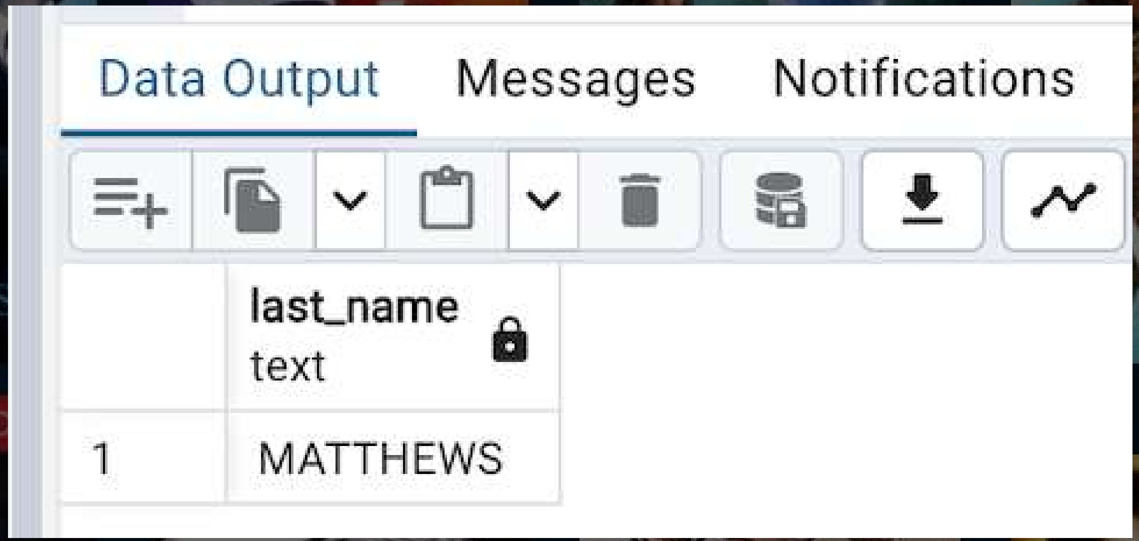


```
     count
     bigint
1        24
```

From the support team, you get a request to find out the last name of the customer with the first name Erica.



```
SELECT
last_name
FROM customer
WHERE First_name = 'ERICA'
```

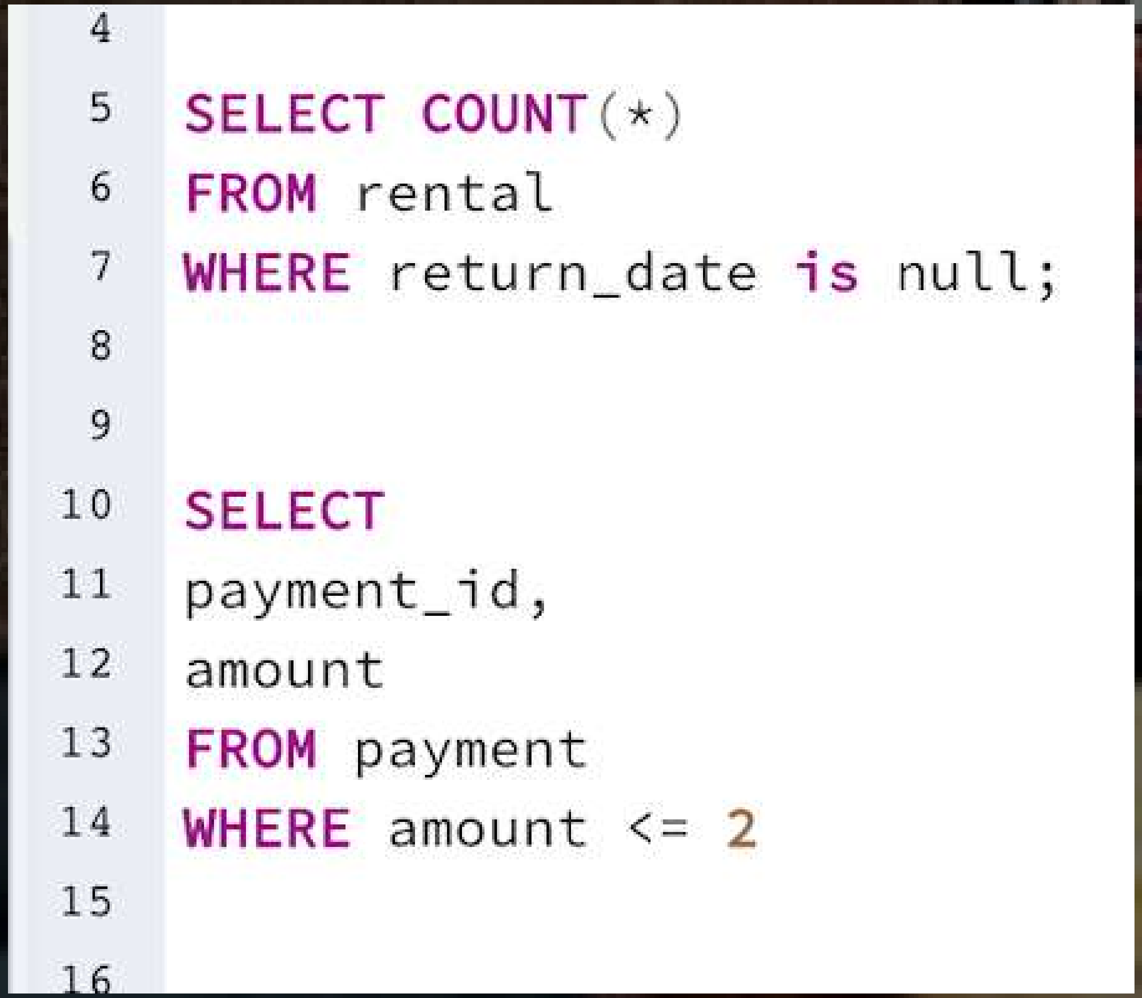


```
     last_name
     text
1    MATTHEWS
```

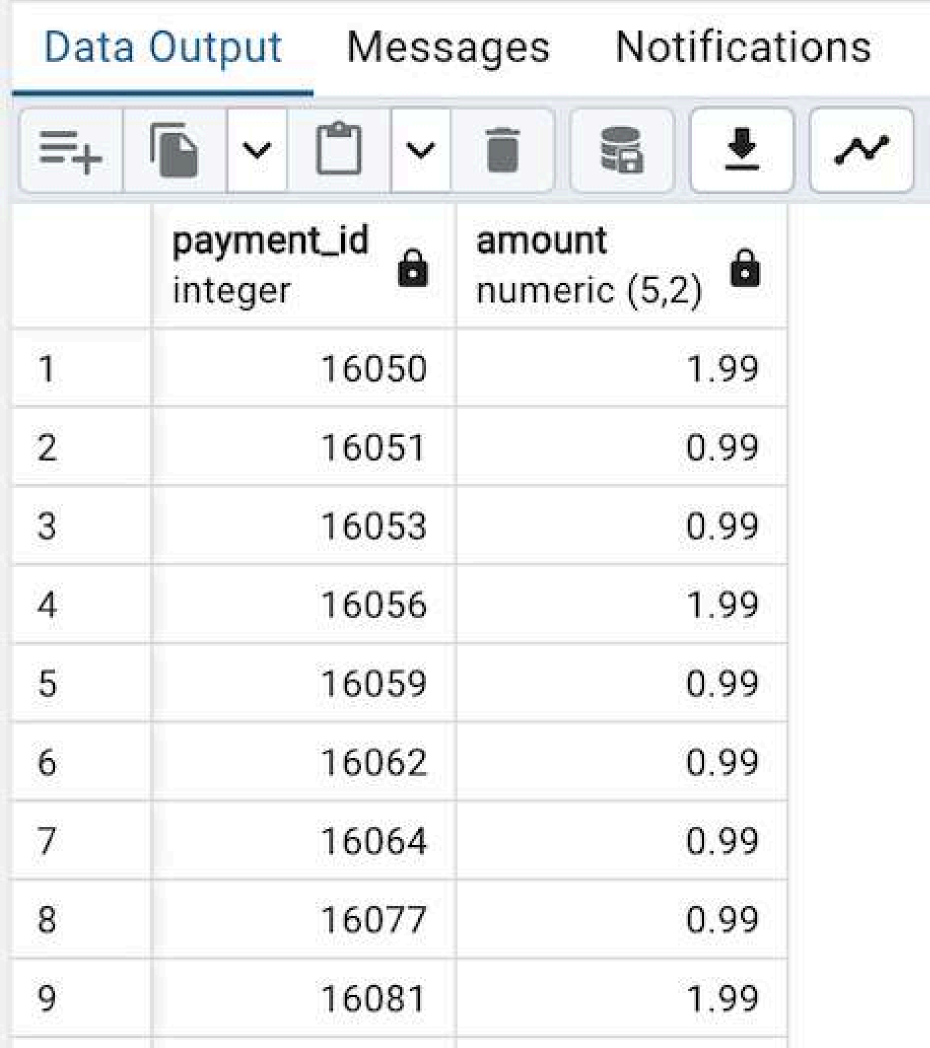
You receive a request from the inventory manager inquiring about the number of rentals that have not yet been returned.

Secondly, The sales manager requests a list of all payment IDs where the amount is $2 or less.

Additionally, they require a list that includes each payment ID along with its corresponding amount.

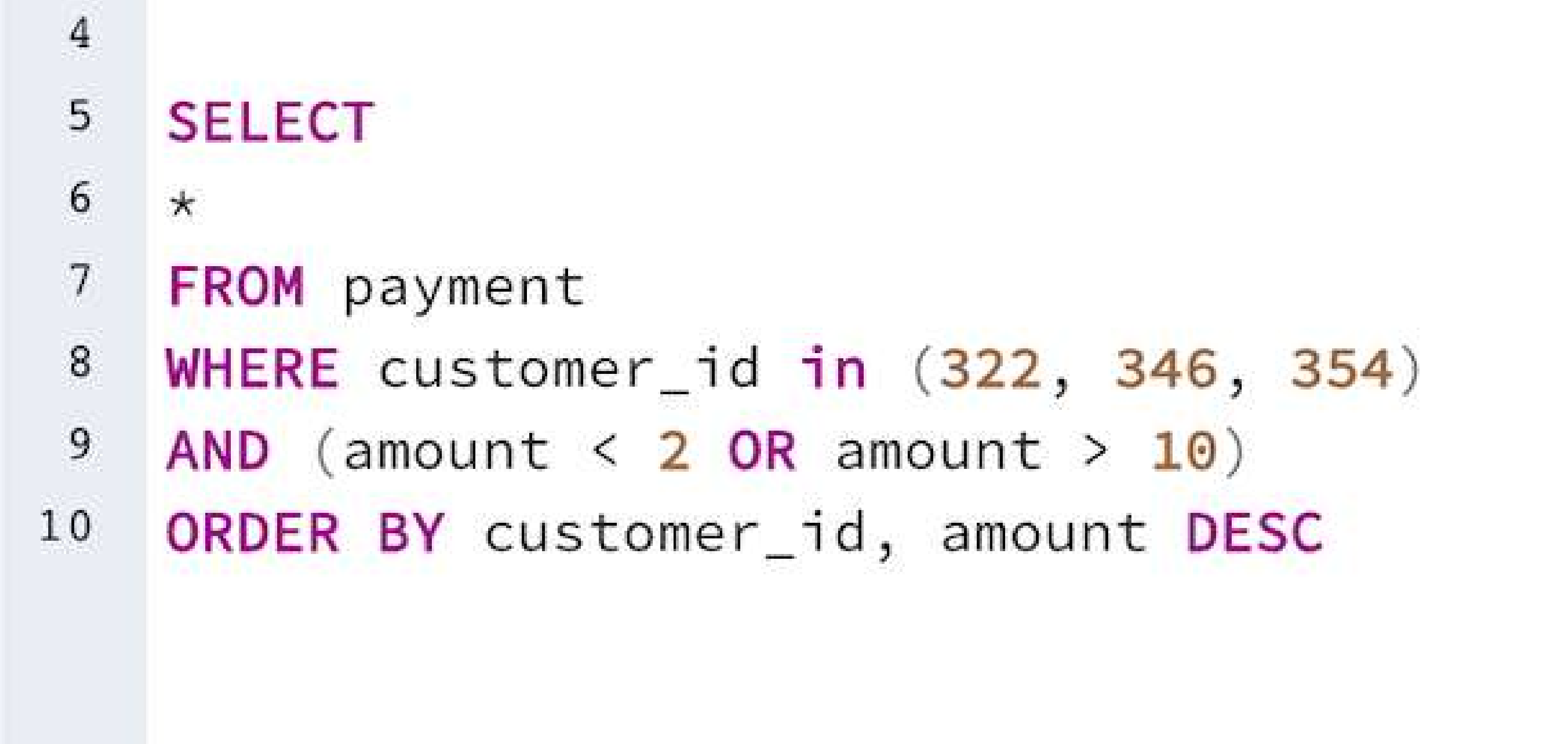


```
SELECT COUNT(*)
FROM rental
WHERE return_date is null;


SELECT
payment_id,
amount
FROM payment
WHERE amount <= 2

```

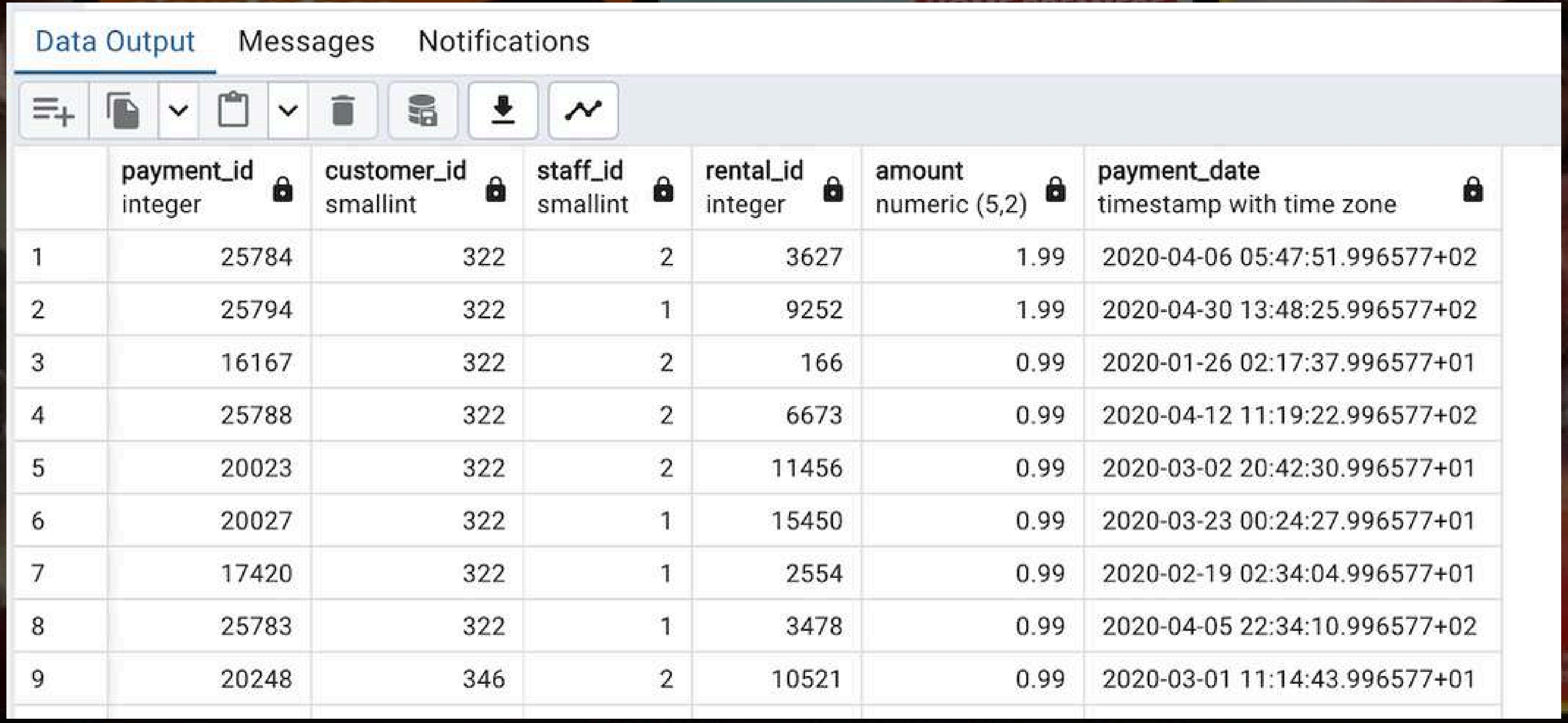


	payment_id integer	amount numeric (5,2)
1	16050	1.99
2	16051	0.99
3	16053	0.99
4	16056	1.99
5	16059	0.99
6	16062	0.99
7	16064	0.99
8	16077	0.99
9	16081	1.99

The support manager inquires about a list detailing payments made by customers 322, 346, and 354. Specifically, they seek amounts for these customers that are either less than $2 or greater than $10. Additionally, the list should be ordered by customer number in ascending order, and then order it by amount in descending order.

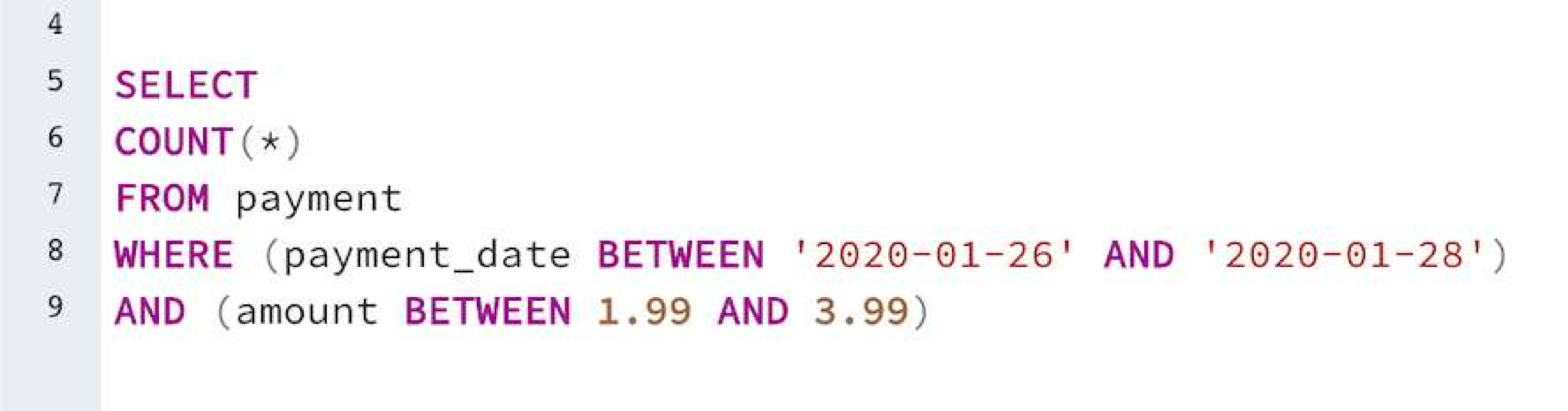


```sql

SELECT
*
FROM payment
WHERE customer_id in (322, 346, 354)
AND (amount < 2 OR amount > 10)
ORDER BY customer_id, amount DESC
```

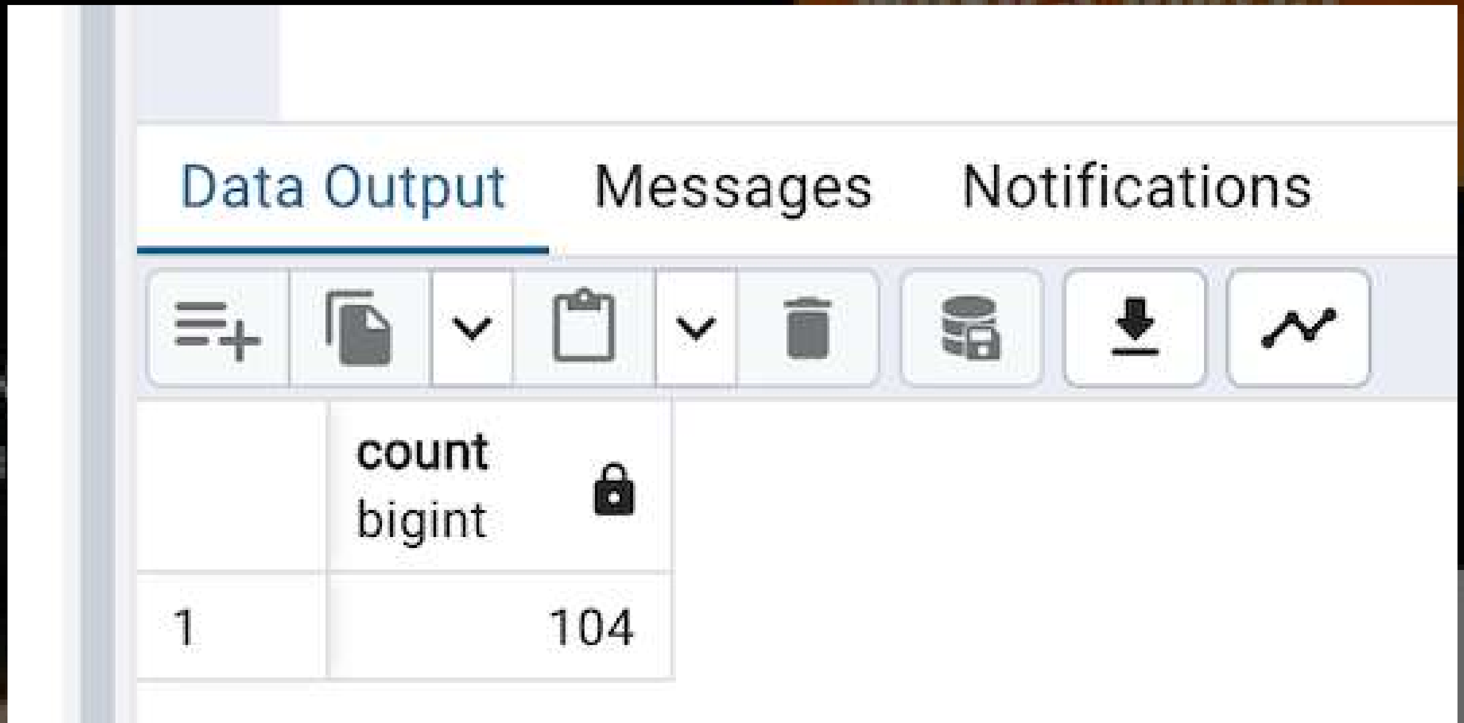


	payment_id integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp with time zone
1	25784	322	2	3627	1.99	2020-04-06 05:47:51.996577+02
2	25794	322	1	9252	1.99	2020-04-30 13:48:25.996577+02
3	16167	322	2	166	0.99	2020-01-26 02:17:37.996577+01
4	25788	322	2	6673	0.99	2020-04-12 11:19:22.996577+02
5	20023	322	2	11456	0.99	2020-03-02 20:42:30.996577+01
6	20027	322	1	15450	0.99	2020-03-23 00:24:27.996577+01
7	17420	322	1	2554	0.99	2020-02-19 02:34:04.996577+01
8	25783	322	1	3478	0.99	2020-04-05 22:34:10.996577+02
9	20248	346	2	10521	0.99	2020-03-01 11:14:43.996577+01

The manager has identified faulty payments and we now need to determine the number of affected transactions. Specifically, we need to find out how many payments were made on January 26 and 27, 2020, with amounts ranging from $1.99 to $3.99.

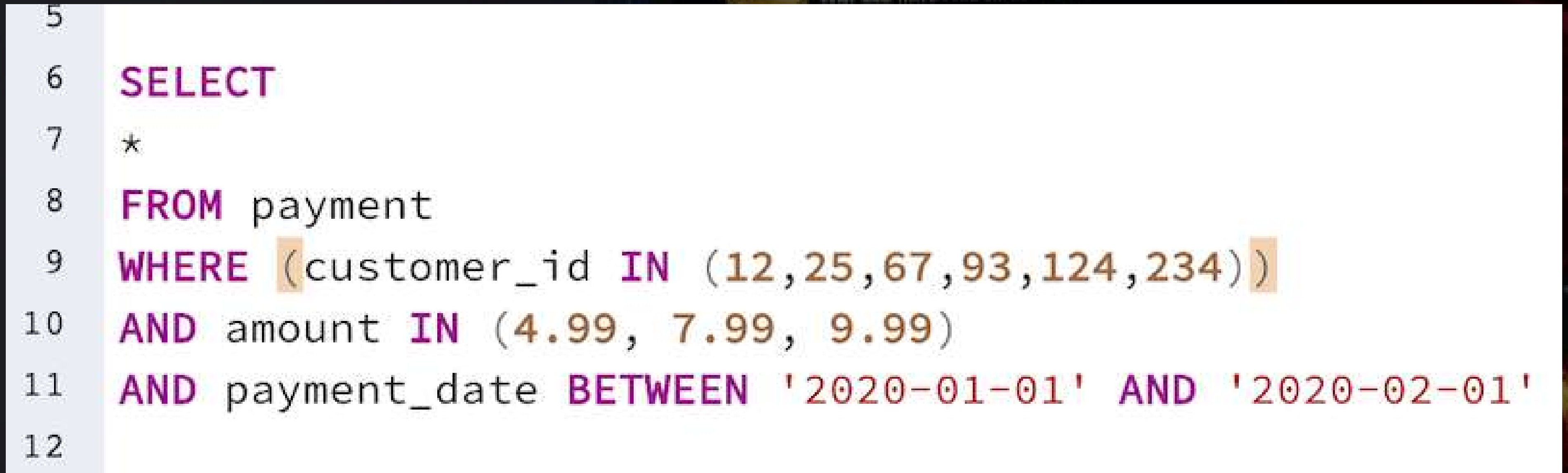


```sql

SELECT
COUNT(*)
FROM payment
WHERE (payment_date BETWEEN '2020-01-26' AND '2020-01-28')
AND (amount BETWEEN 1.99 AND 3.99)
```

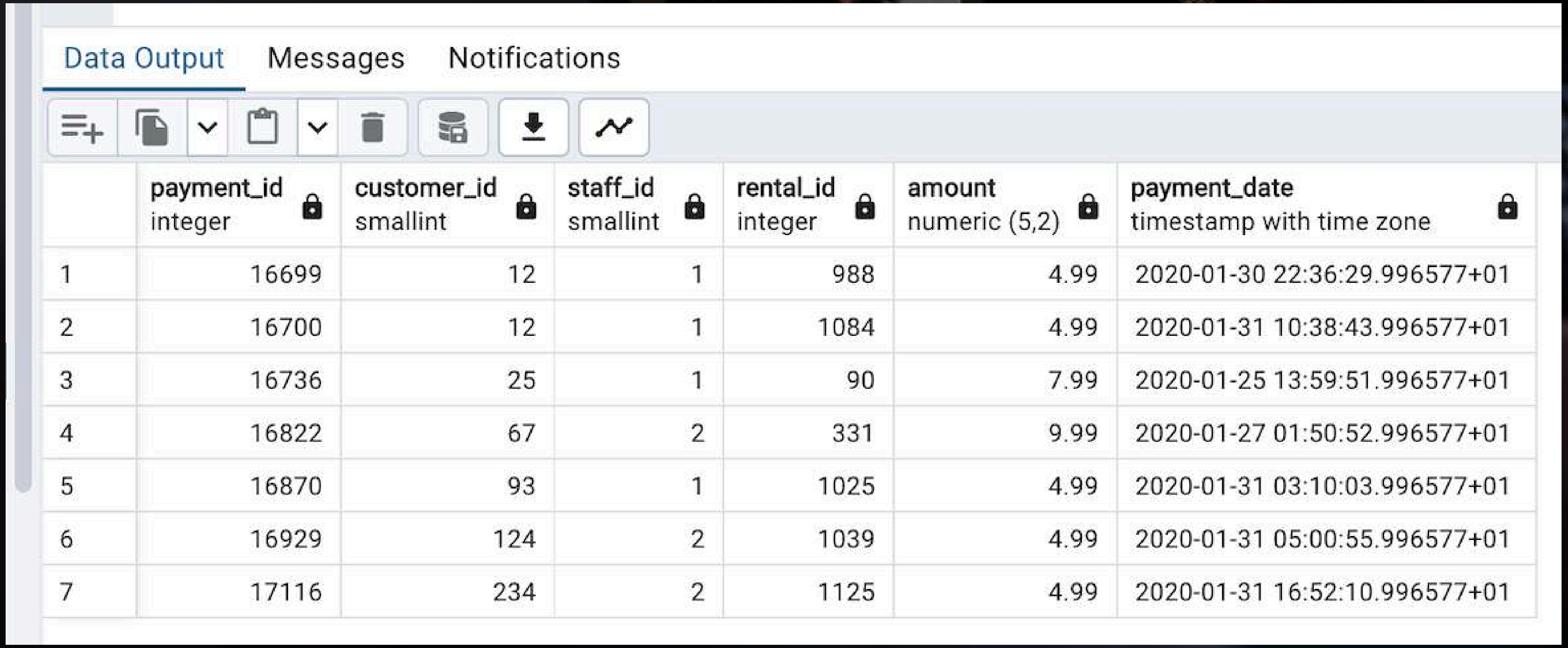


```
   count
   bigint
1     104
```

The support manager has approached you again, informing you of six complaints from customers regarding their payments. You now need to investigate these payments for the customers with the following IDs: 12, 25, 67, 93, 124, and 234. Specifically, you need to look into payments of these customers with amounts of $4.99, $7.99, and $9.99, all occurring in January 2020.

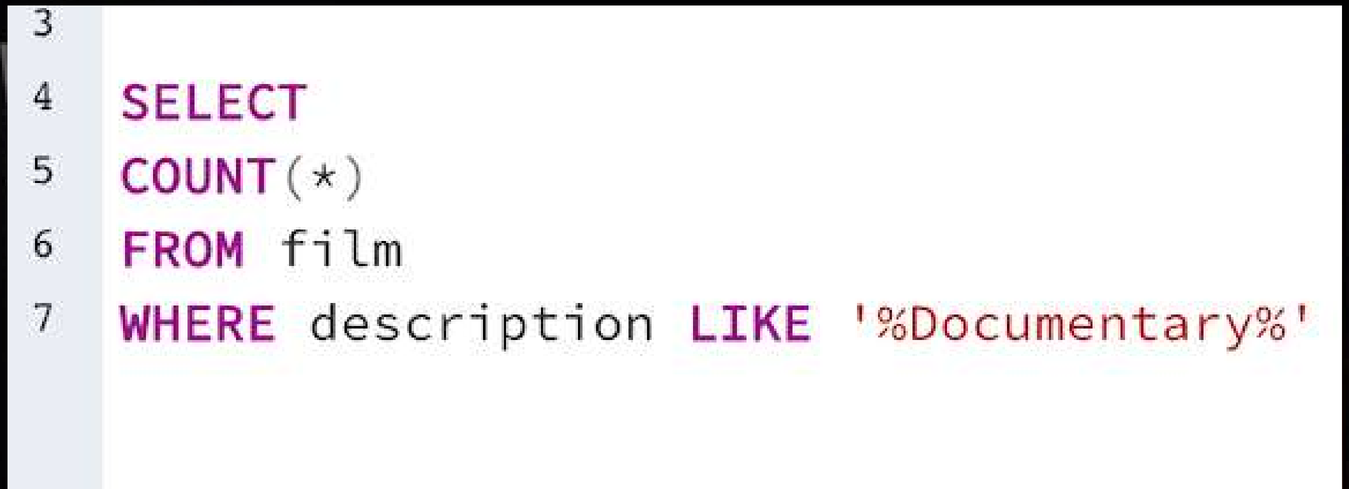


```sql

SELECT
*
FROM payment
WHERE (customer_id IN (12,25,67,93,124,234))
AND amount IN (4.99, 7.99, 9.99)
AND payment_date BETWEEN '2020-01-01' AND '2020-02-01'

```

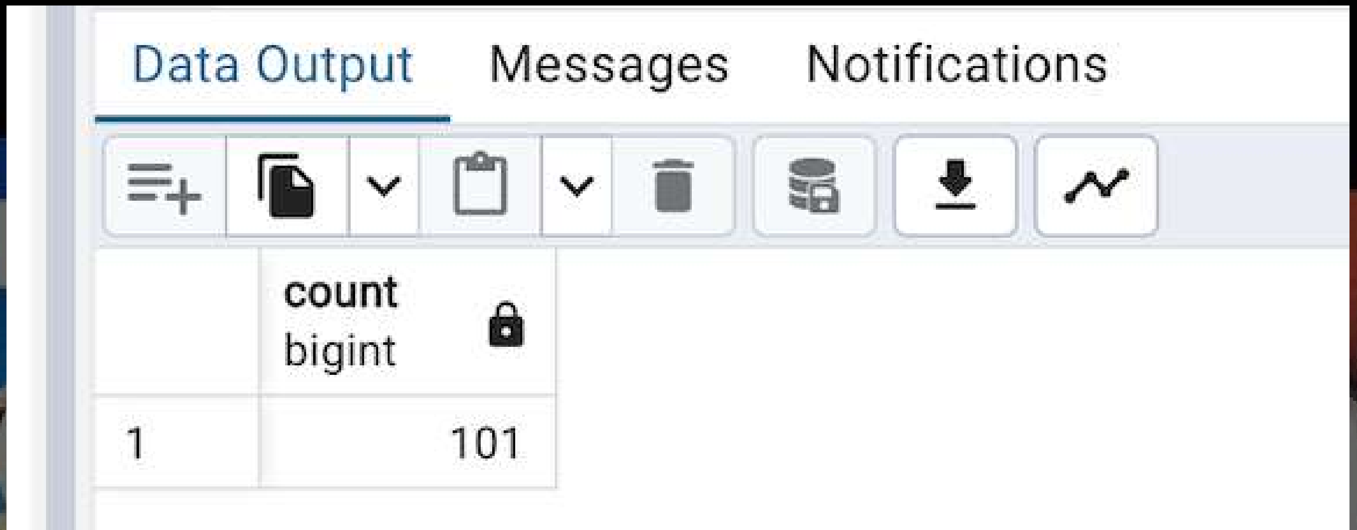


	payment_id integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp with time zone
1	16699	12	1	988	4.99	2020-01-30 22:36:29.996577+01
2	16700	12	1	1084	4.99	2020-01-31 10:38:43.996577+01
3	16736	25	1	90	7.99	2020-01-25 13:59:51.996577+01
4	16822	67	2	331	9.99	2020-01-27 01:50:52.996577+01
5	16870	93	1	1025	4.99	2020-01-31 03:10:03.996577+01
6	16929	124	2	1039	4.99	2020-01-31 05:00:55.996577+01
7	17116	234	2	1125	4.99	2020-01-31 16:52:10.996577+01

The inventory manager requires assistance once more; this time, we need to determine the number of movies that have the word 'documentary' in their description.

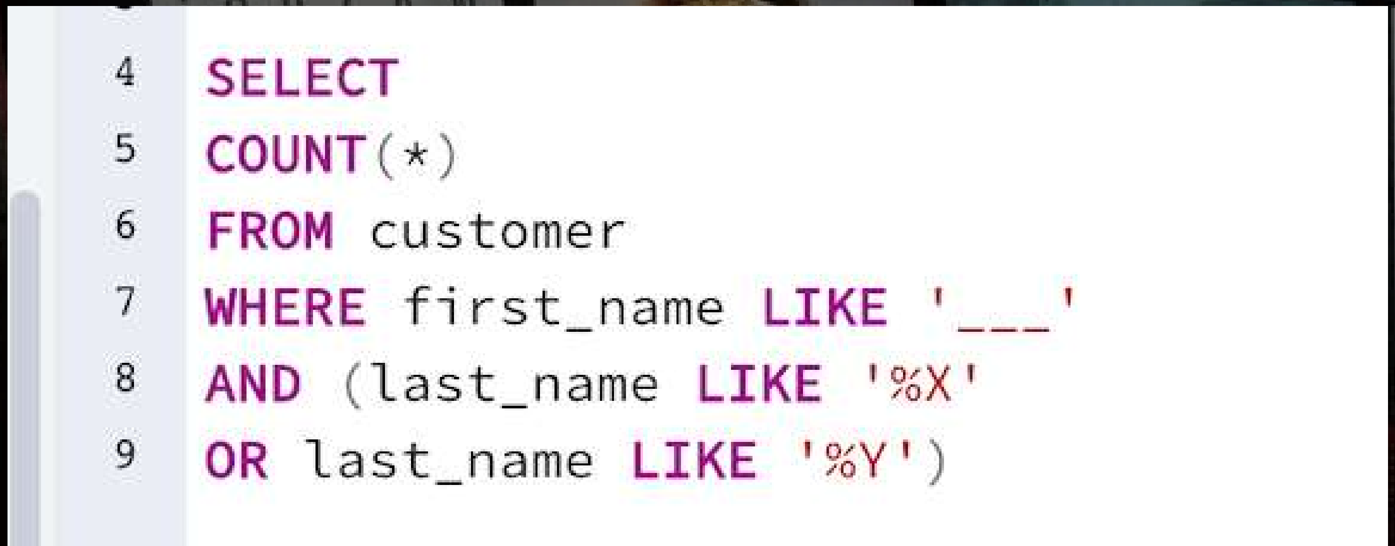


```
SELECT
COUNT(*)
FROM film
WHERE description LIKE '%Documentary%'
```

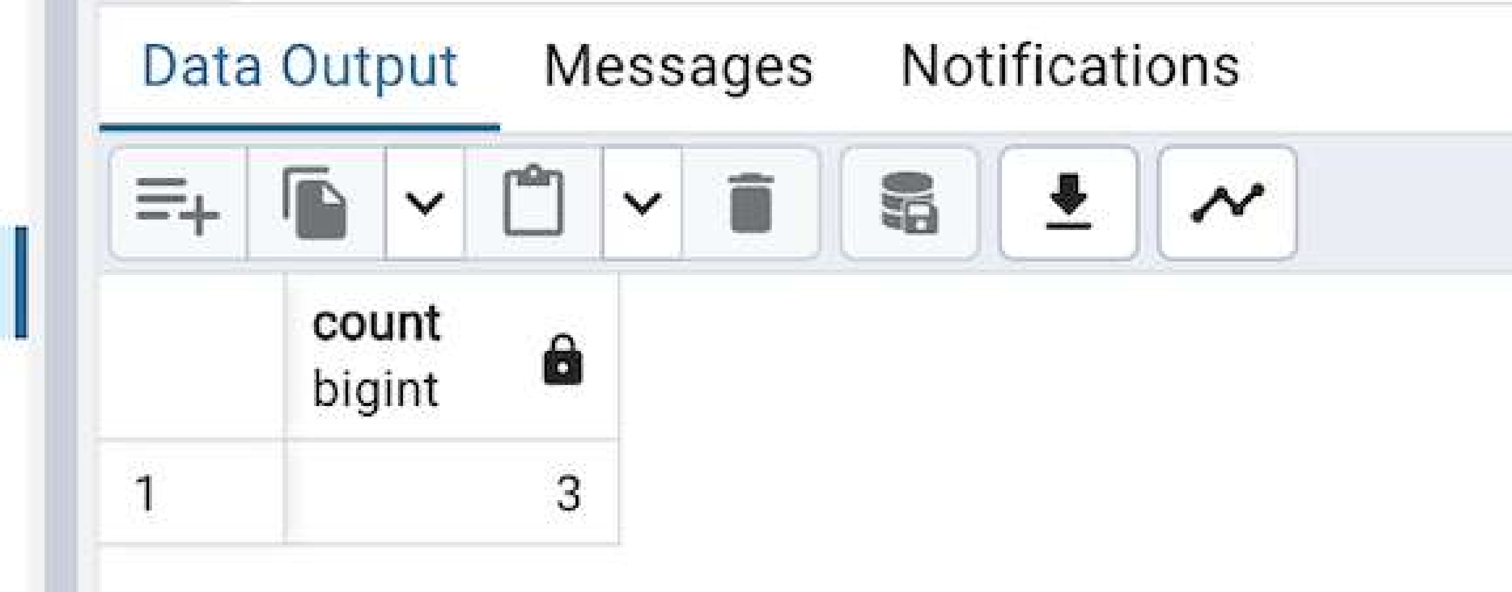


```
   count
   bigint
1     101
```

You need to determine the number of customers in the database whose first name is three letters long and whose last name ends with either an X or a Y.



```
SELECT
COUNT(*)
FROM customer
WHERE first_name LIKE '___'
AND (last_name LIKE '%X'
OR last_name LIKE '%Y')
```

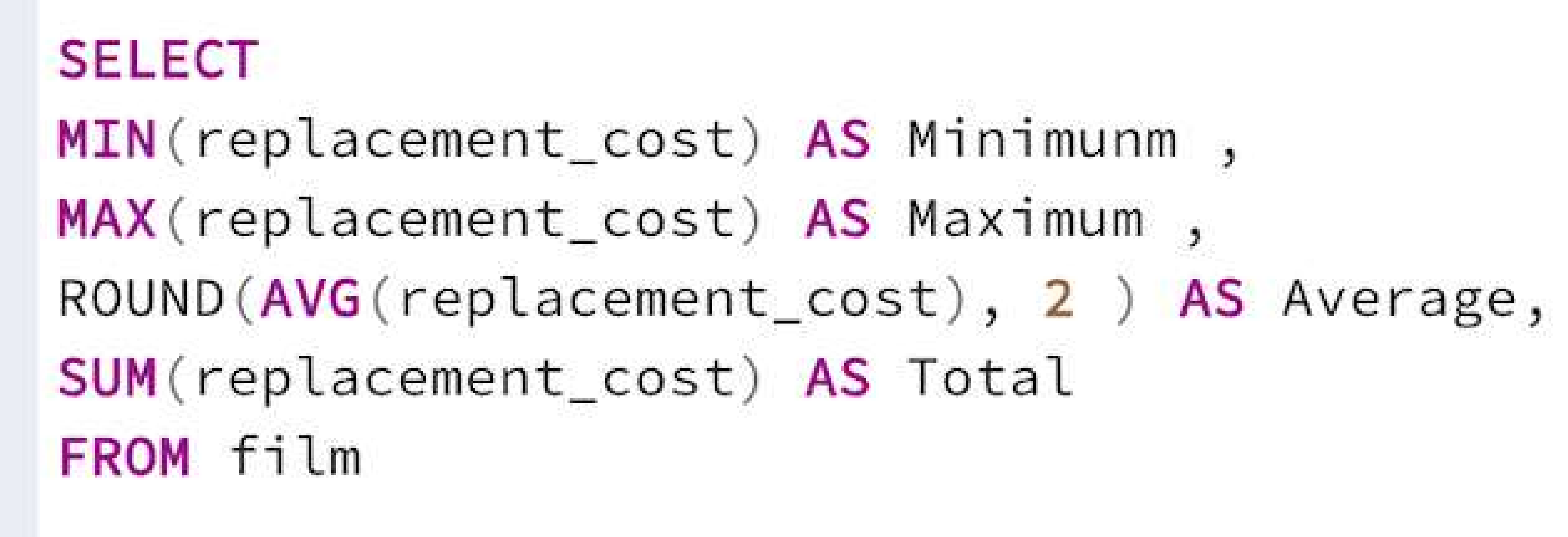


```
   count
   bigint
1       3
```

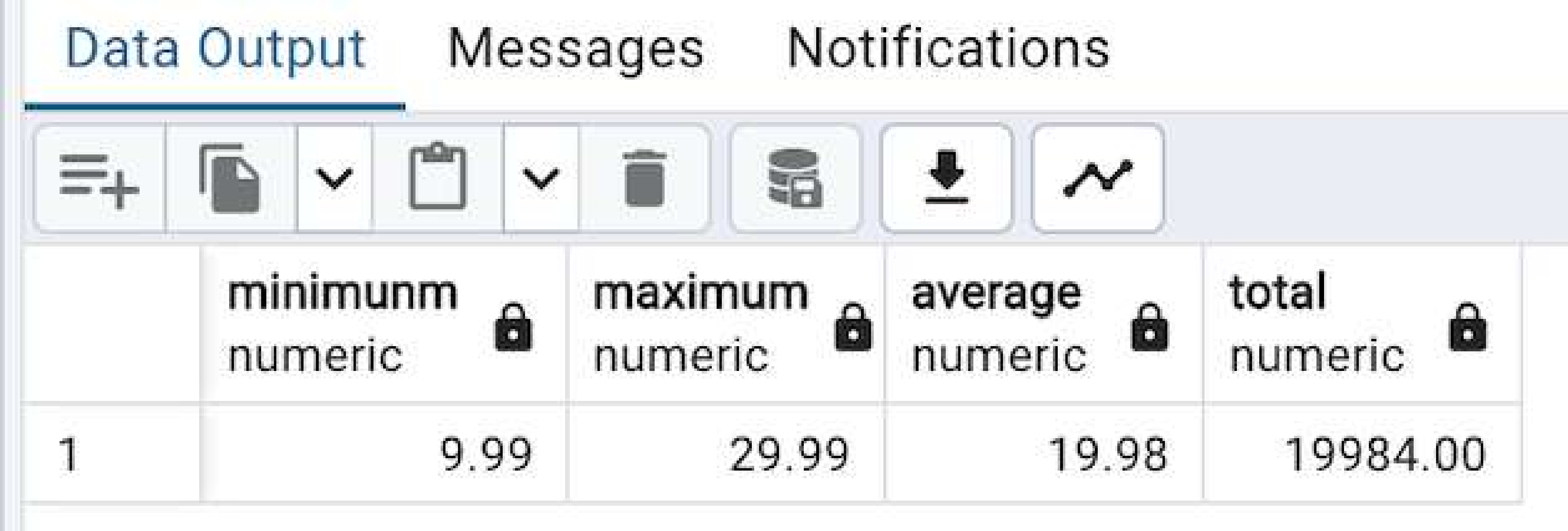
The manager now wants to gather more information about the films in THE FILM CORNER store.

You are requested to write a query to determine the replacement costs of these films.

Specifically, you should find the minimum, maximum, and average (rounded to two decimal places) replacement costs, as well as the total sum.

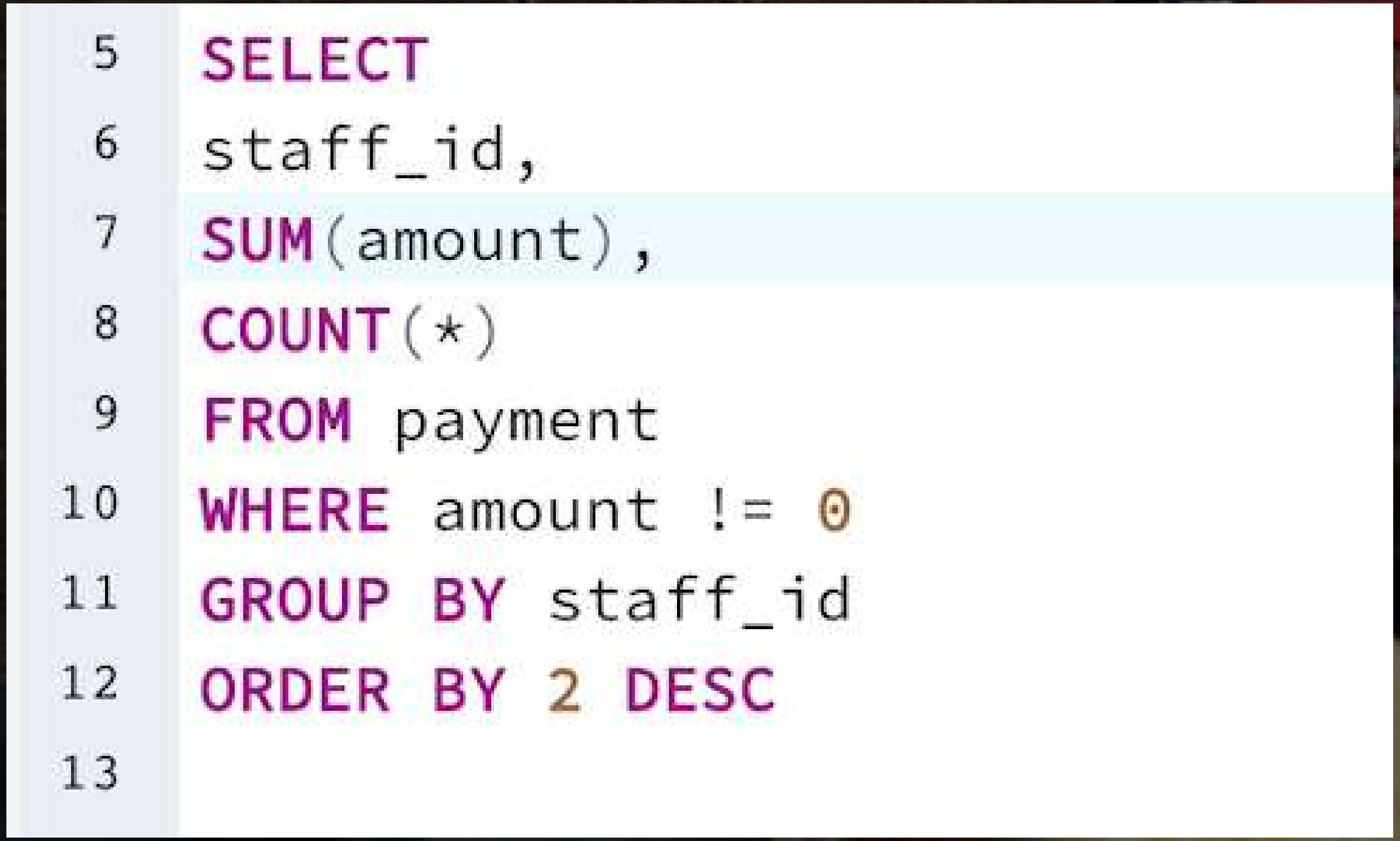


```
SELECT
MIN(replacement_cost) AS Minimunm ,
MAX(replacement_cost) AS Maximum ,
ROUND(AVG(replacement_cost), 2 ) AS Average,
SUM(replacement_cost) AS Total
FROM film
```

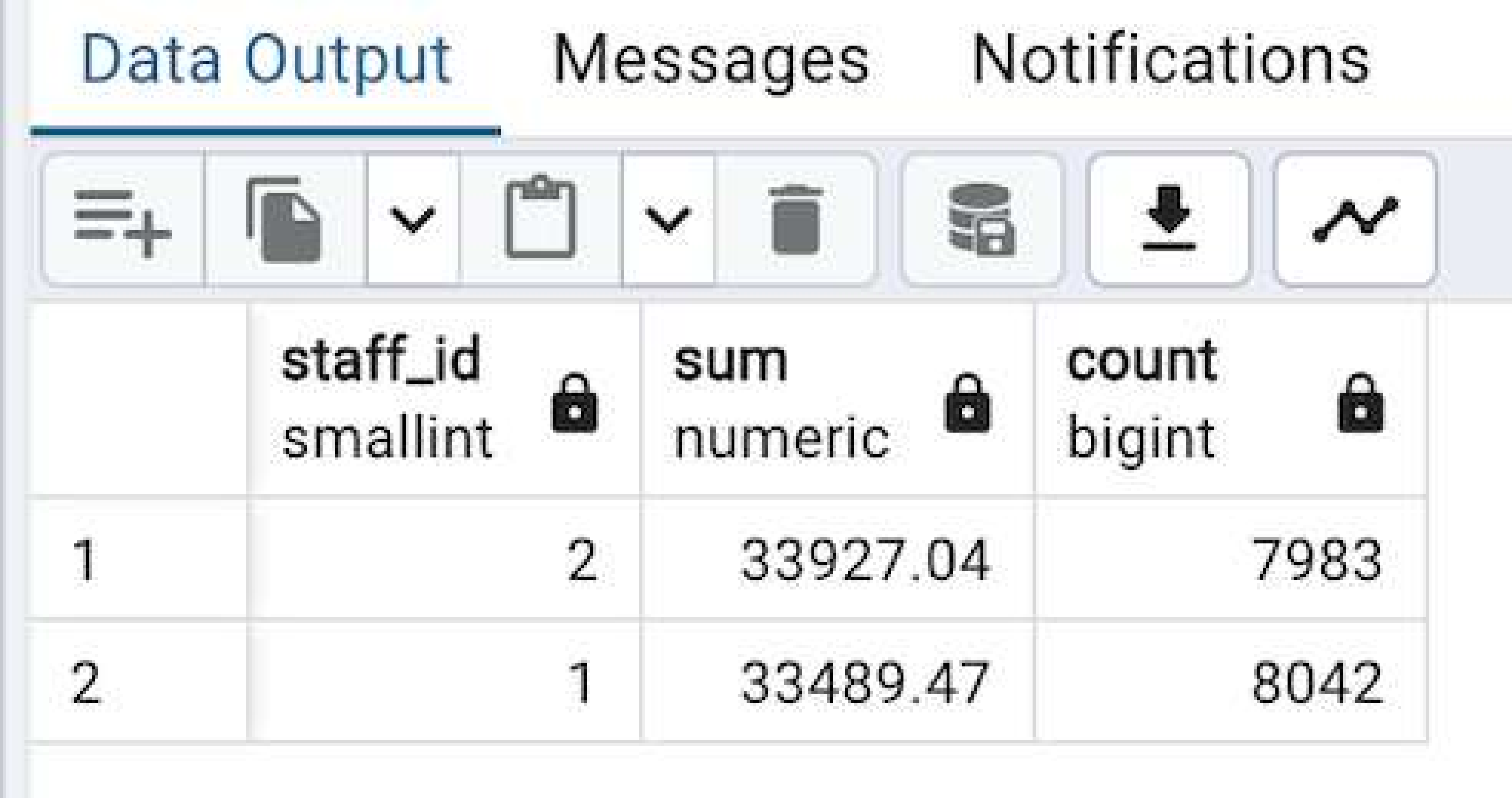


	minimunm numeric	maximum numeric	average numeric	total numeric
1	9.99	29.99	19.98	19984.00

The manager wants you to determine which of the two employees, identified by their staff IDs, is responsible for more payments. Additionally, you should identify which of these employees has handled a higher total payment amount. Finally, assess how these figures would change if you excluded payments with amounts equal to zero.

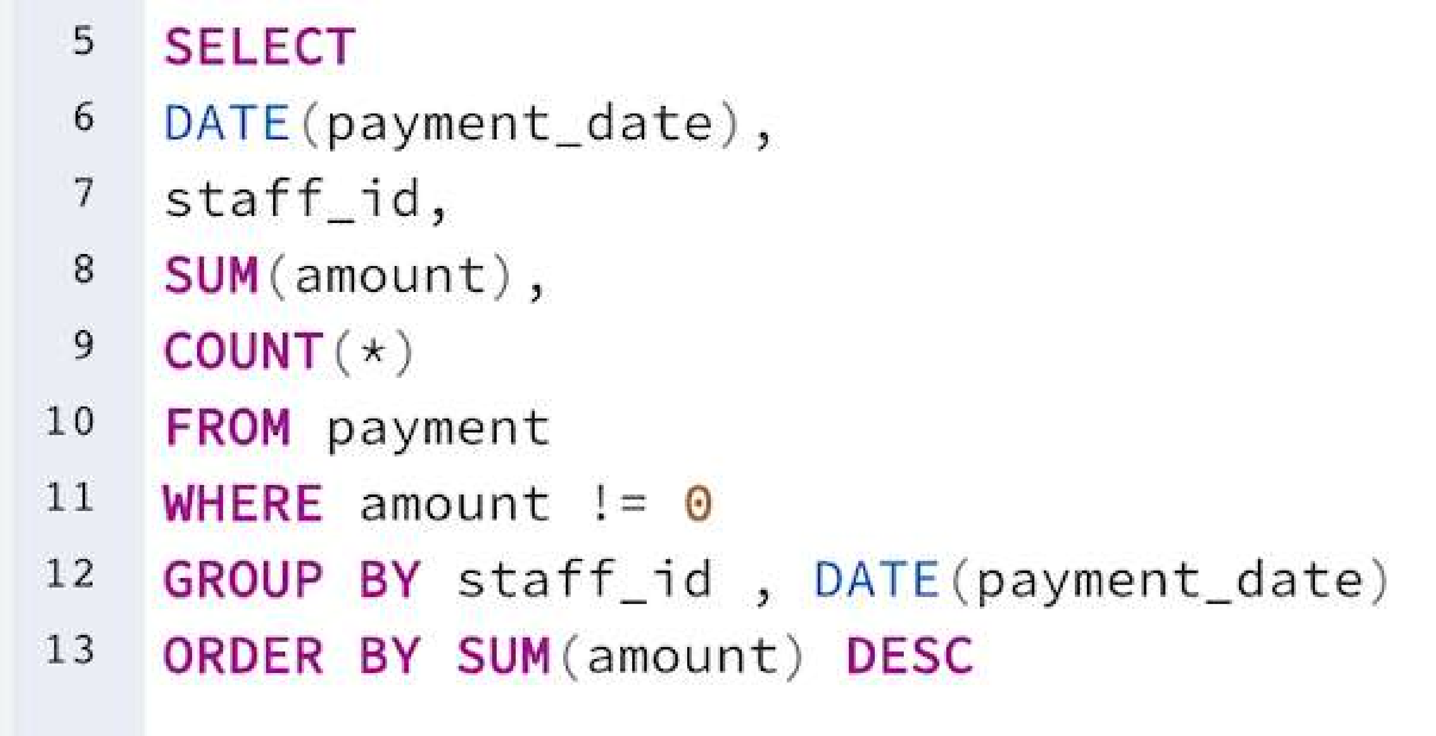


```sql
SELECT
staff_id,
SUM(amount),
COUNT(*)
FROM payment
WHERE amount != 0
GROUP BY staff_id
ORDER BY 2 DESC

```

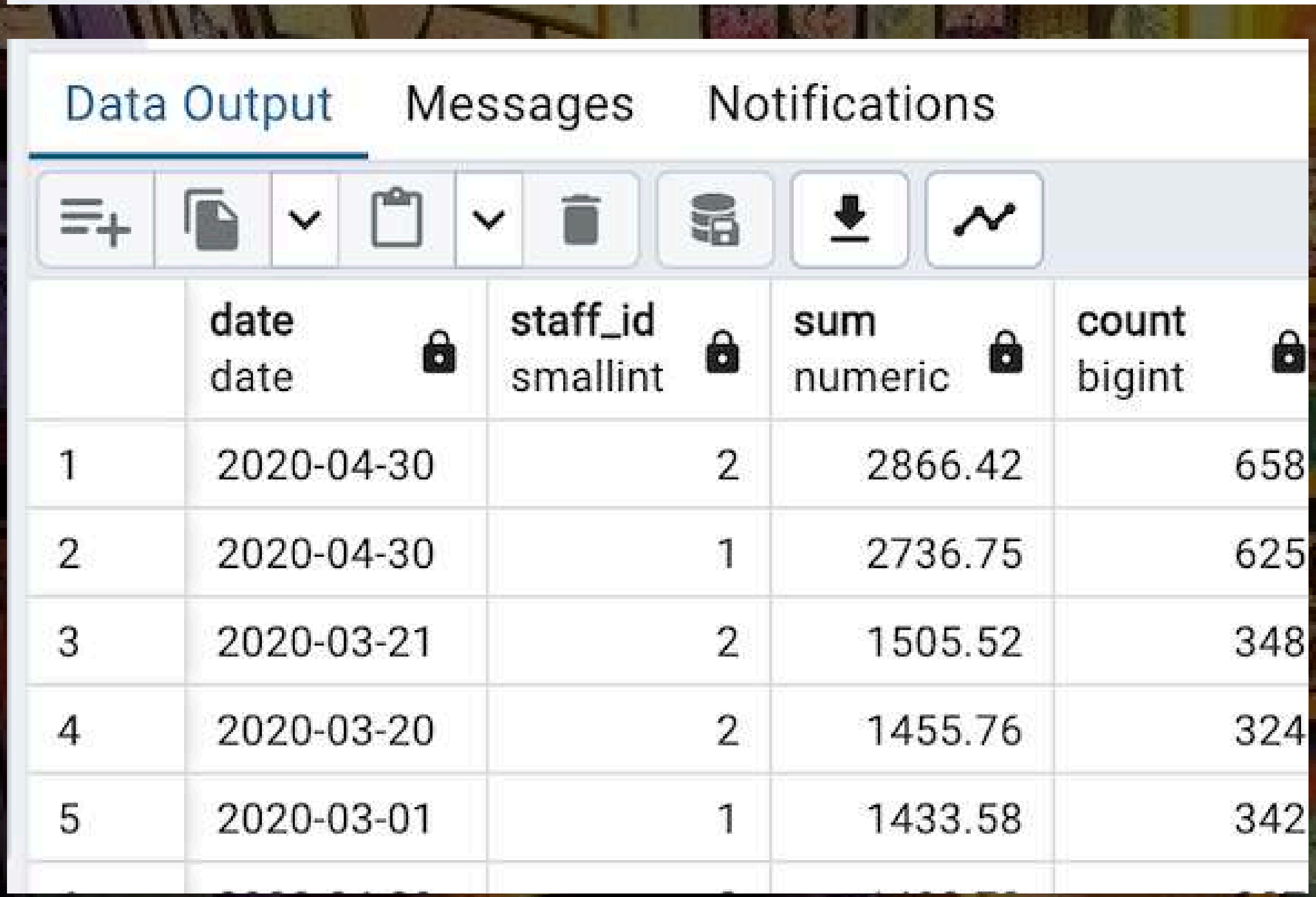


	staff_id smallint	sum numeric	count bigint
1	2	33927.04	7983
2	1	33489.47	8042

"We need to determine which of the two employees achieved the highest sales amount in a single day. Additionally, we need to find out which of these employees recorded the most sales transactions in a single day. Note that payments with amounts equal to zero should not be included in this analysis."

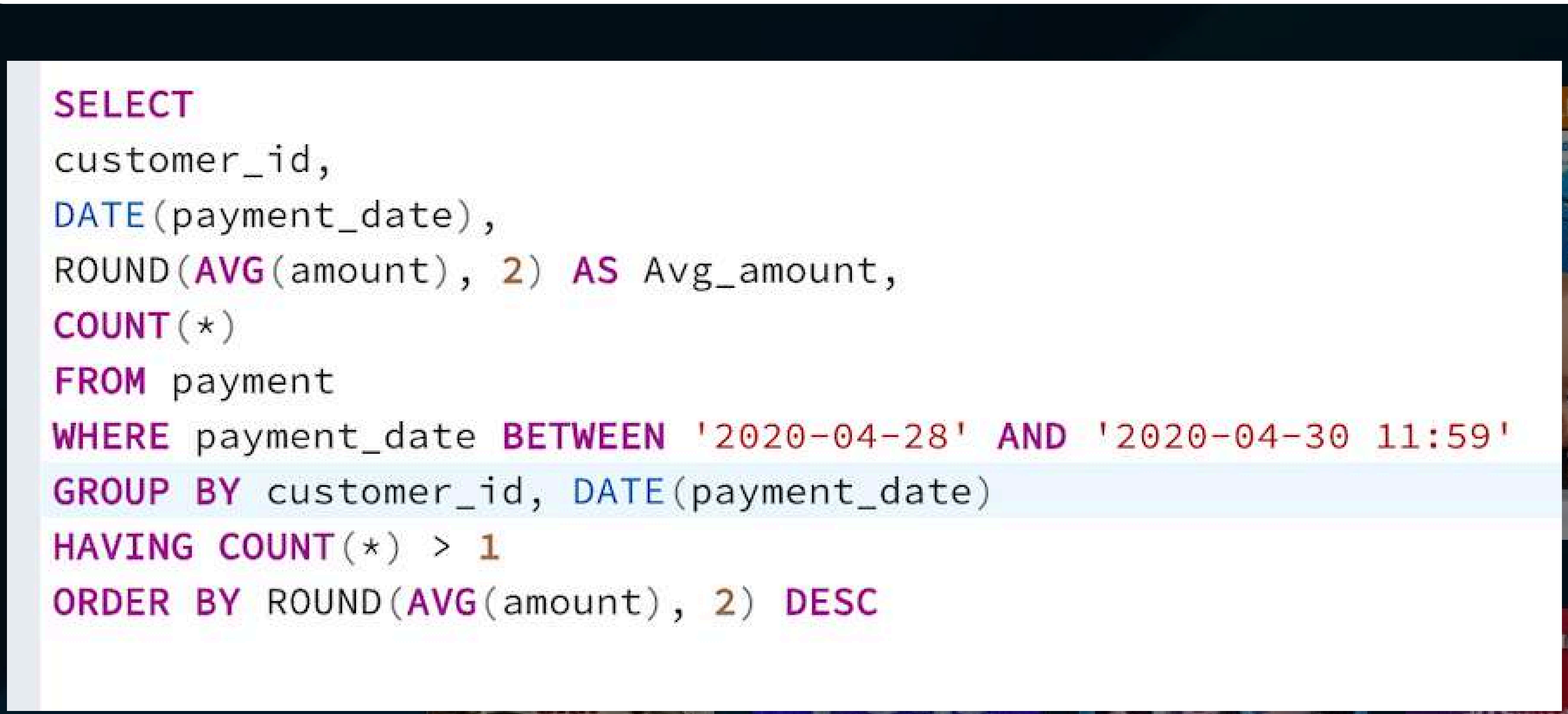


```sql
SELECT
DATE(payment_date),
staff_id,
SUM(amount),
COUNT(*)
FROM payment
WHERE amount != 0
GROUP BY staff_id , DATE(payment_date)
ORDER BY SUM(amount) DESC
```

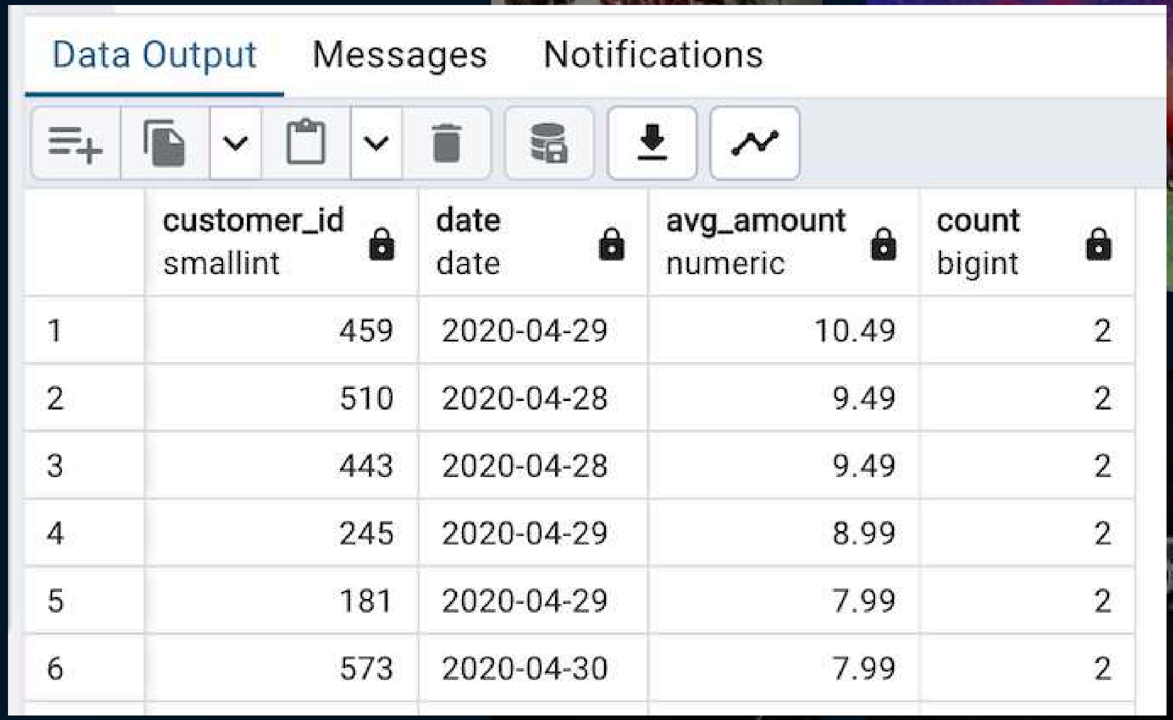


	date date	staff_id smallint	sum numeric	count bigint
1	2020-04-30	2	2866.42	658
2	2020-04-30	1	2736.75	625
3	2020-03-21	2	1505.52	348
4	2020-03-20	2	1455.76	324
5	2020-03-01	1	1433.58	342

The manager has identified April 28, 29, and 30, 2020, as days with exceptionally high revenue and wants to focus exclusively on these dates. The task is to calculate the average payment amount, grouping by customer and date. Additionally, restrict the results to customers and dates with more than one payment each, and sort the results in descending order of the average amount."

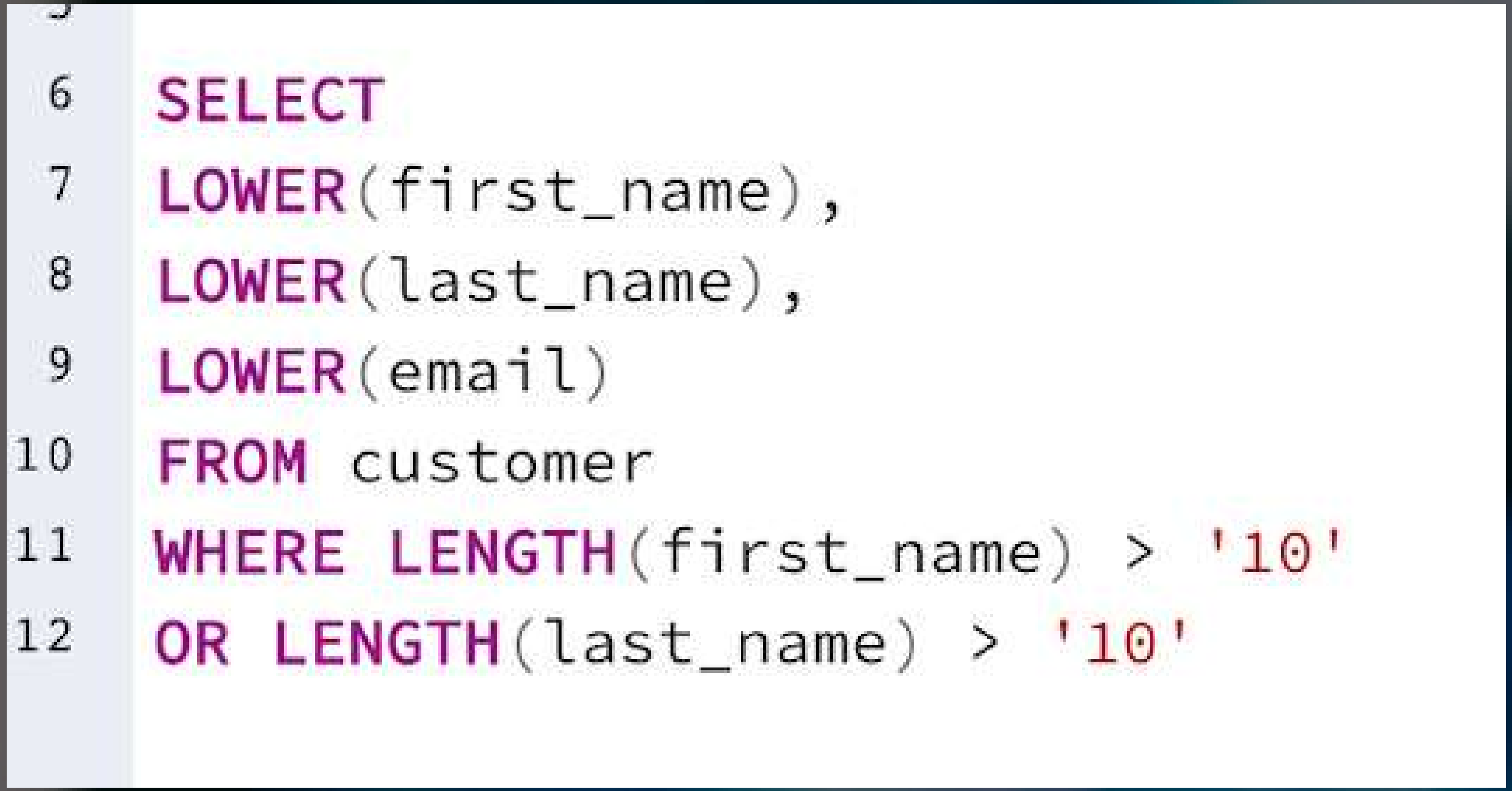


```sql
SELECT
customer_id,
DATE(payment_date),
ROUND(AVG(amount), 2) AS Avg_amount,
COUNT(*)
FROM payment
WHERE payment_date BETWEEN '2020-04-28' AND '2020-04-30 11:59'
GROUP BY customer_id, DATE(payment_date)
HAVING COUNT(*) > 1
ORDER BY ROUND(AVG(amount), 2) DESC
```

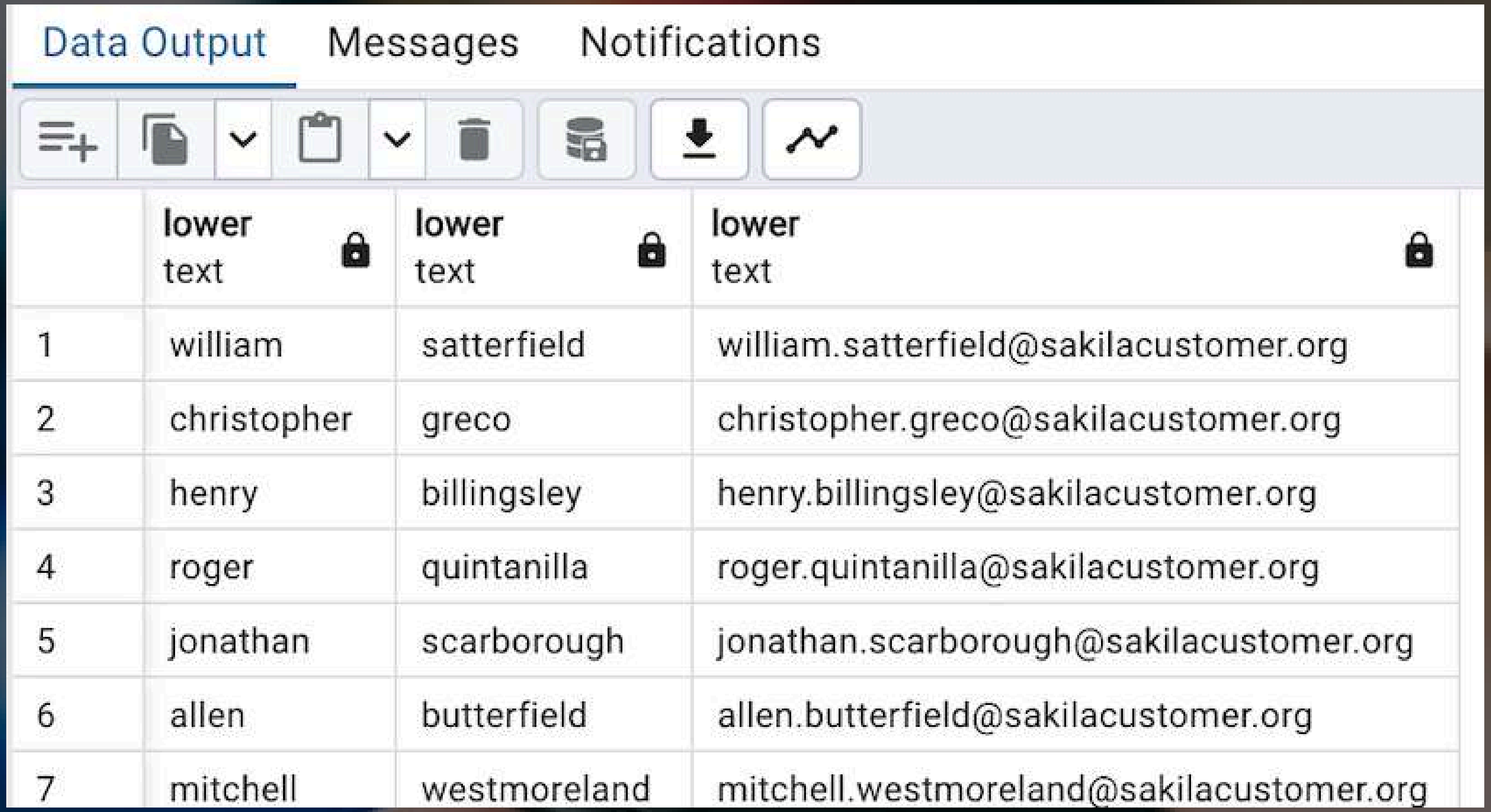


	customer_id smallint	date date	avg_amount numeric	count bigint
1	459	2020-04-29	10.49	2
2	510	2020-04-28	9.49	2
3	443	2020-04-28	9.49	2
4	245	2020-04-29	8.99	2
5	181	2020-04-29	7.99	2
6	573	2020-04-30	7.99	2

There was an issue with the company's email system that prevented emails from being sent when either the first name or the last name had more than ten characters. Therefore, your task is to create a list of all customers who meet this criteria, ensuring that both the names and email addresses are in lowercase letters.

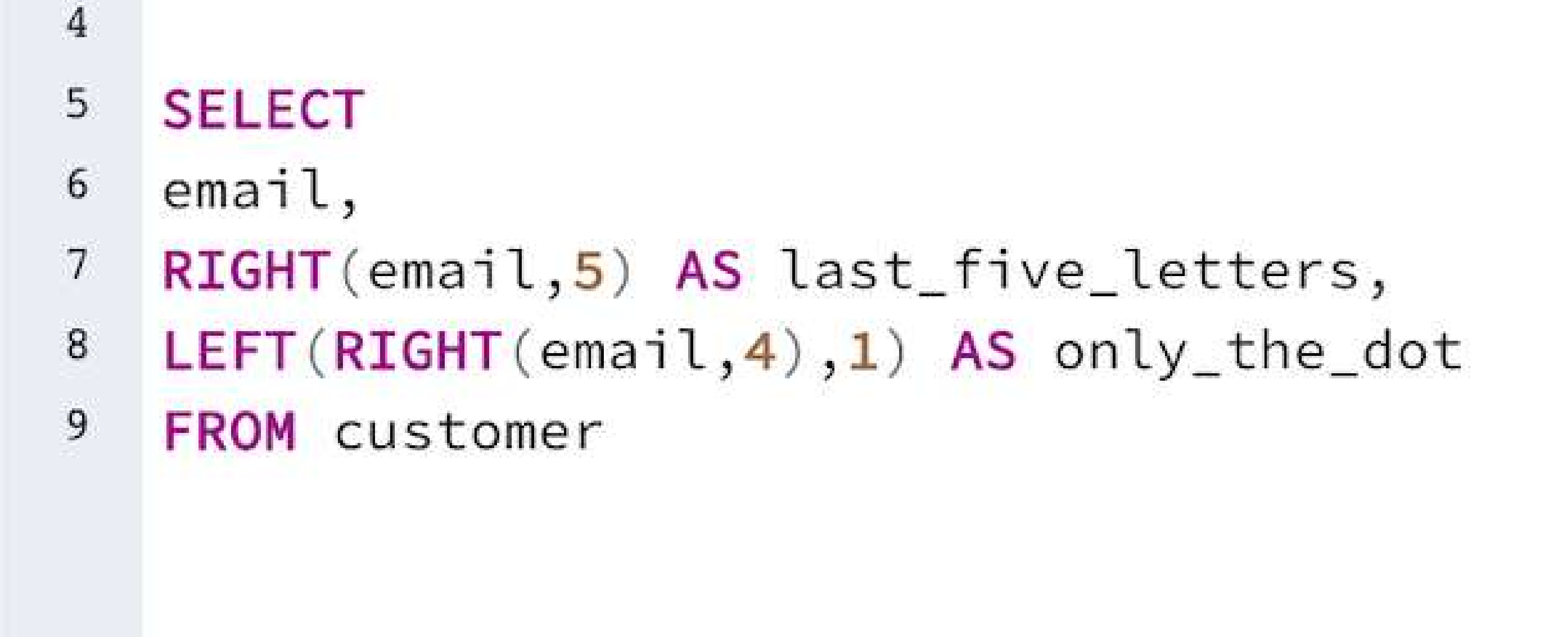


```
SELECT
LOWER(first_name),
LOWER(last_name),
LOWER(email)
FROM customer
WHERE LENGTH(first_name) > '10'
OR LENGTH(last_name) > '10'
```

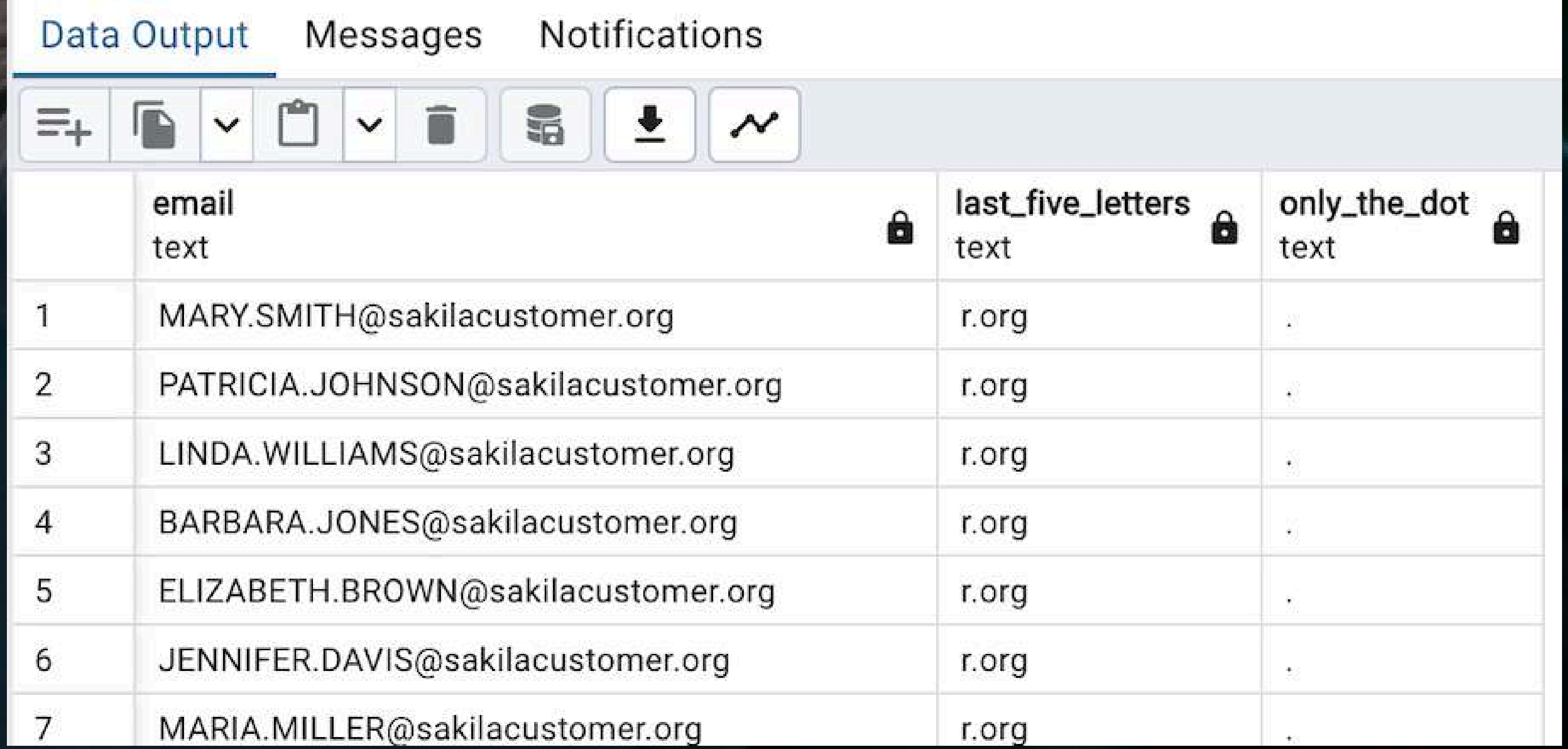


	lower text	lower text	lower text
1	william	satterfield	william.satterfield@sakilacustomer.org
2	christopher	greco	christopher.greco@sakilacustomer.org
3	henry	billingsley	henry.billingsley@sakilacustomer.org
4	roger	quintanilla	roger.quintanilla@sakilacustomer.org
5	jonathan	scarborough	jonathan.scarborough@sakilacustomer.org
6	allen	butterfield	allen.butterfield@sakilacustomer.org
7	mitchell	westmoreland	mitchell.westmoreland@sakilacustomer.org

Your task is to extract the last five characters of the email addresses. Additionally, note that the email addresses always end with ".org." You need to extract just the dot from this ending. How can you accomplish this?

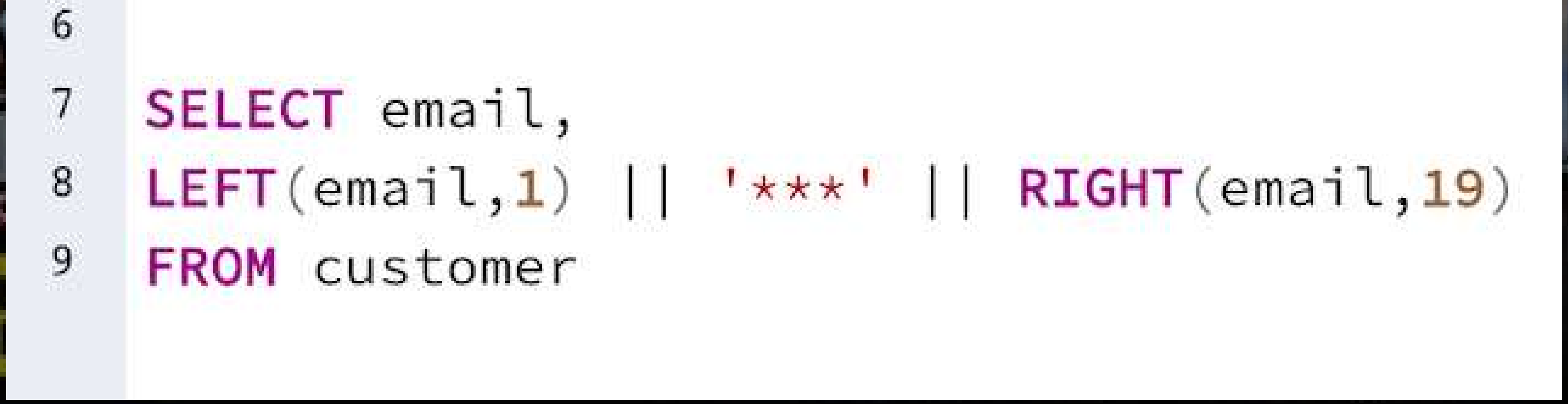


```sql

SELECT
email,
RIGHT(email,5) AS last_five_letters,
LEFT(RIGHT(email,4),1) AS only_the_dot
FROM customer
```

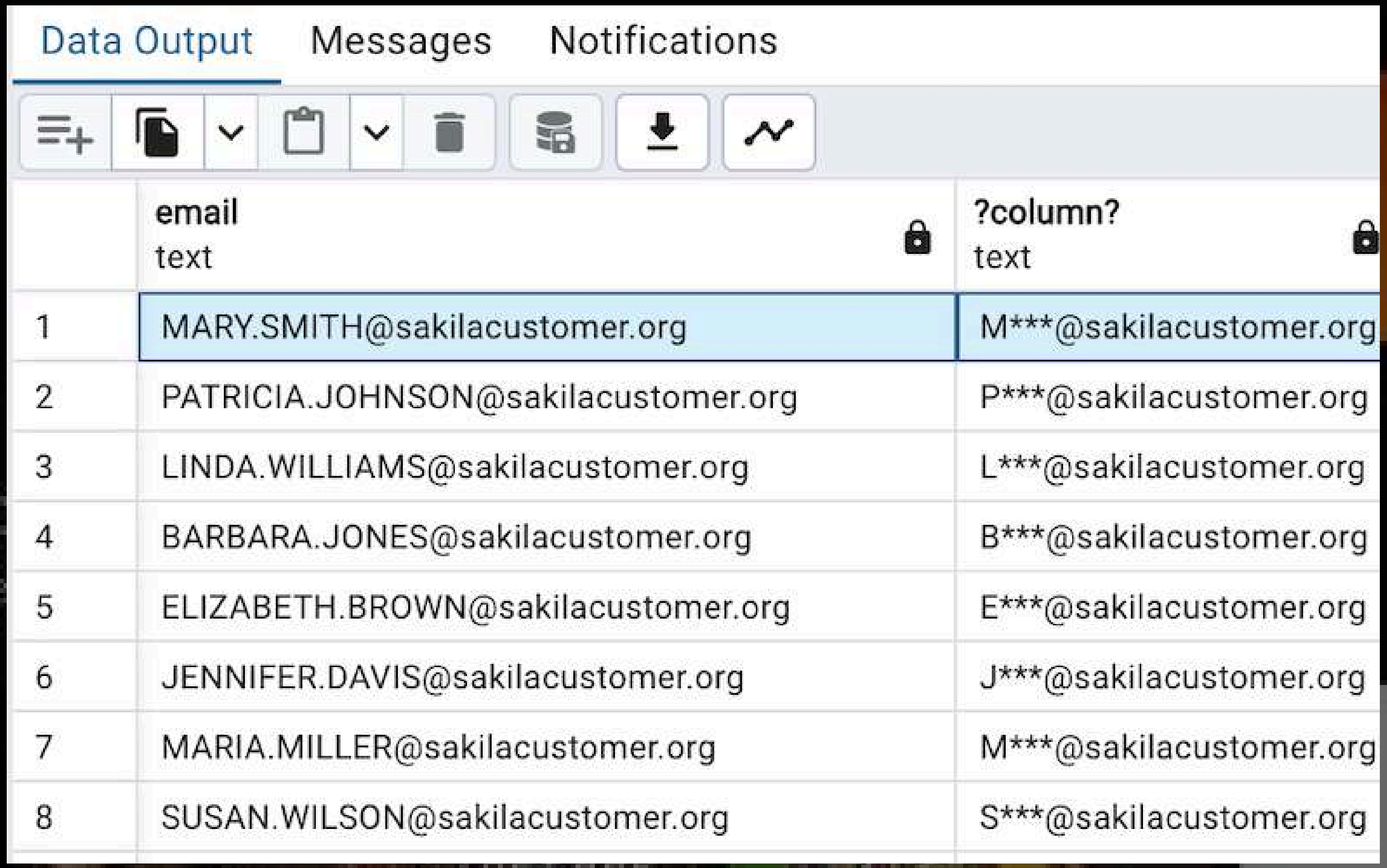


	email text	last_five_letters text	only_the_dot text
1	MARY.SMITH@sakilacustomer.org	r.org	.
2	PATRICIA.JOHNSON@sakilacustomer.org	r.org	.
3	LINDA.WILLIAMS@sakilacustomer.org	r.org	.
4	BARBARA.JONES@sakilacustomer.org	r.org	.
5	ELIZABETH.BROWN@sakilacustomer.org	r.org	.
6	JENNIFER.DAVIS@sakilacustomer.org	r.org	.
7	MARIA.MILLER@sakilacustomer.org	r.org	.

You should now anonymize the email addresses by creating a list in the following format:
the first character of the email address, followed by three asterisks, and then the email provider.
For example, "M***@sakilacustomer.org.

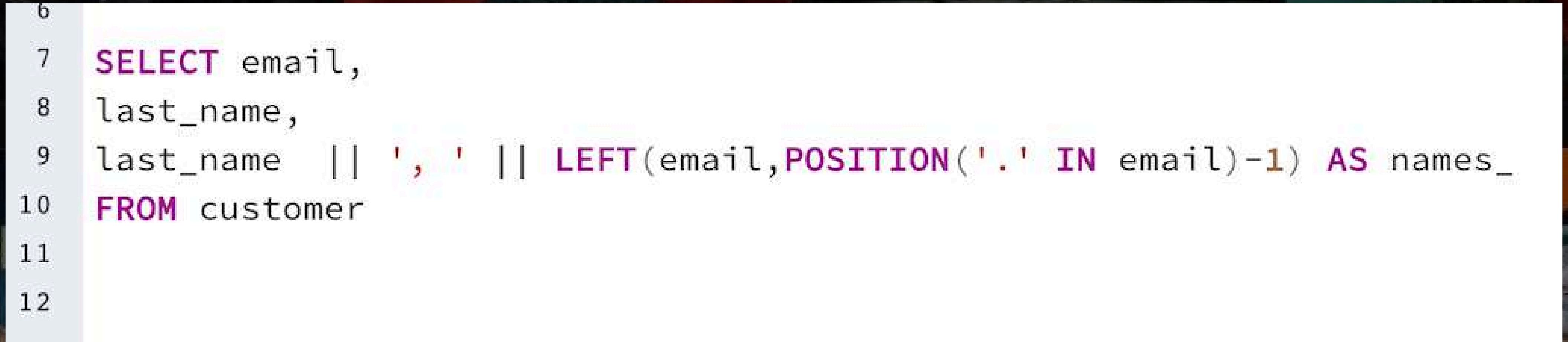


```
SELECT email,
LEFT(email,1) || '***' || RIGHT(email,19)
FROM customer
```

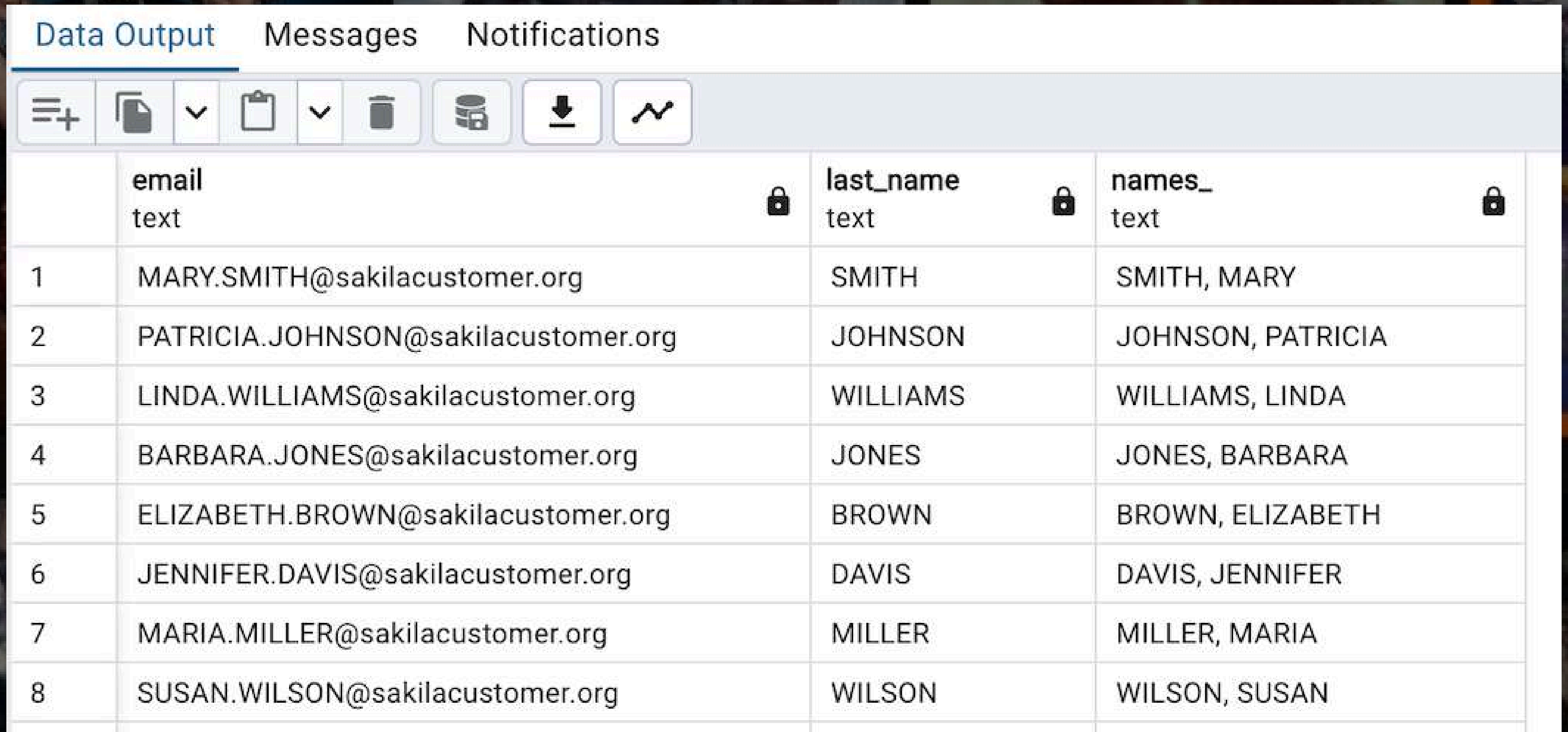


	email text	?column? text
1	MARY.SMITH@sakilacustomer.org	M***@sakilacustomer.org
2	PATRICIA.JOHNSON@sakilacustomer.org	P***@sakilacustomer.org
3	LINDA.WILLIAMS@sakilacustomer.org	L***@sakilacustomer.org
4	BARBARA.JONES@sakilacustomer.org	B***@sakilacustomer.org
5	ELIZABETH.BROWN@sakilacustomer.org	E***@sakilacustomer.org
6	JENNIFER.DAVIS@sakilacustomer.org	J***@sakilacustomer.org
7	MARIA.MILLER@sakilacustomer.org	M***@sakilacustomer.org
8	SUSAN.WILSON@sakilacustomer.org	S***@sakilacustomer.org

In this task, you only have access to the email address and the last name.
You cannot use any other columns. You need to extract the first name from the email address, then concatenate it with the last name you already have.
The output should be formatted as "last name, first name," separated by a comma and a space, with the first name extracted from the email address.

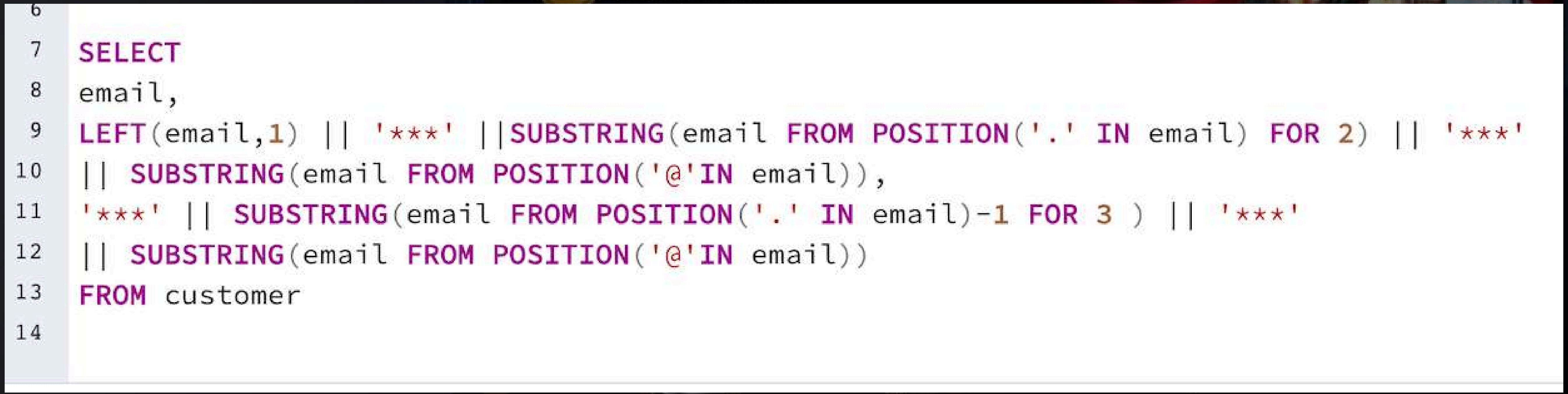


```
SELECT email,
last_name,
last_name  || ', ' || LEFT(email,POSITION('.' IN email)-1) AS names_
FROM customer
```

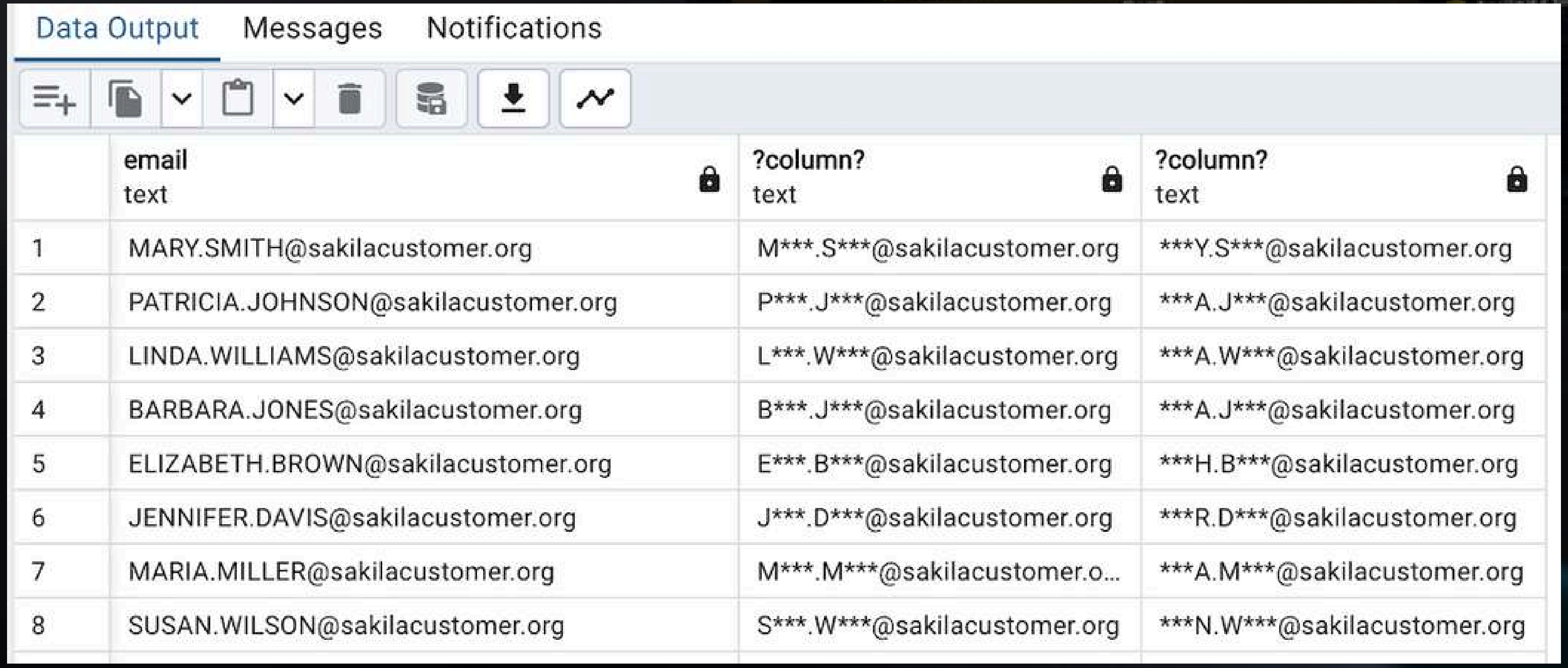


	email text	last_name text	names_ text
1	MARY.SMITH@sakilacustomer.org	SMITH	SMITH, MARY
2	PATRICIA.JOHNSON@sakilacustomer.org	JOHNSON	JOHNSON, PATRICIA
3	LINDA.WILLIAMS@sakilacustomer.org	WILLIAMS	WILLIAMS, LINDA
4	BARBARA.JONES@sakilacustomer.org	JONES	JONES, BARBARA
5	ELIZABETH.BROWN@sakilacustomer.org	BROWN	BROWN, ELIZABETH
6	JENNIFER.DAVIS@sakilacustomer.org	DAVIS	DAVIS, JENNIFER
7	MARIA.MILLER@sakilacustomer.org	MILLER	MILLER, MARIA
8	SUSAN.WILSON@sakilacustomer.org	WILSON	WILSON, SUSAN

In this task, you need to create an anonymized version of the email addresses.
The first format should be like "M***.S***@sakilacustomer.org" or "P***.J***@sakilacustomer.org".
For the second query, the output should be in the format "***Y.S***@sakilacustomer.org"
or "***A.J***@sakilacustomer.org"

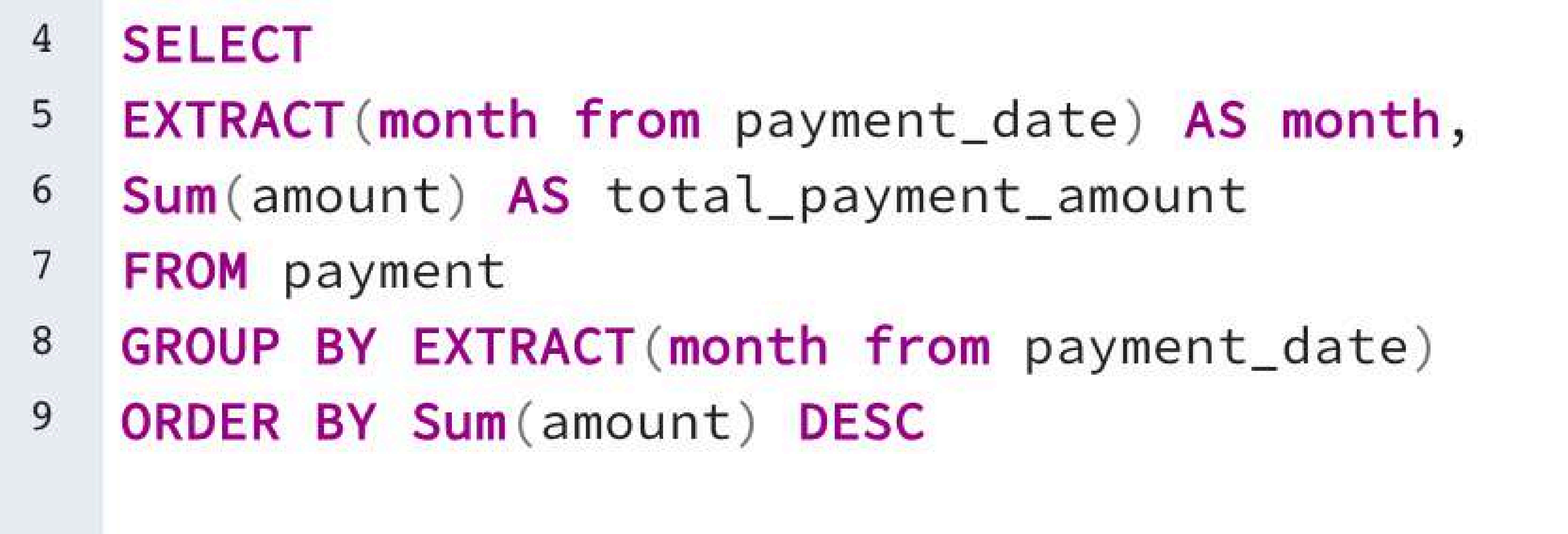


```
SELECT
email,
LEFT(email,1) || '***' ||SUBSTRING(email FROM POSITION('.' IN email) FOR 2) || '***'
|| SUBSTRING(email FROM POSITION('@'IN email)),
'***' || SUBSTRING(email FROM POSITION('.' IN email)-1 FOR 3 ) || '***'
|| SUBSTRING(email FROM POSITION('@'IN email))
FROM customer

```

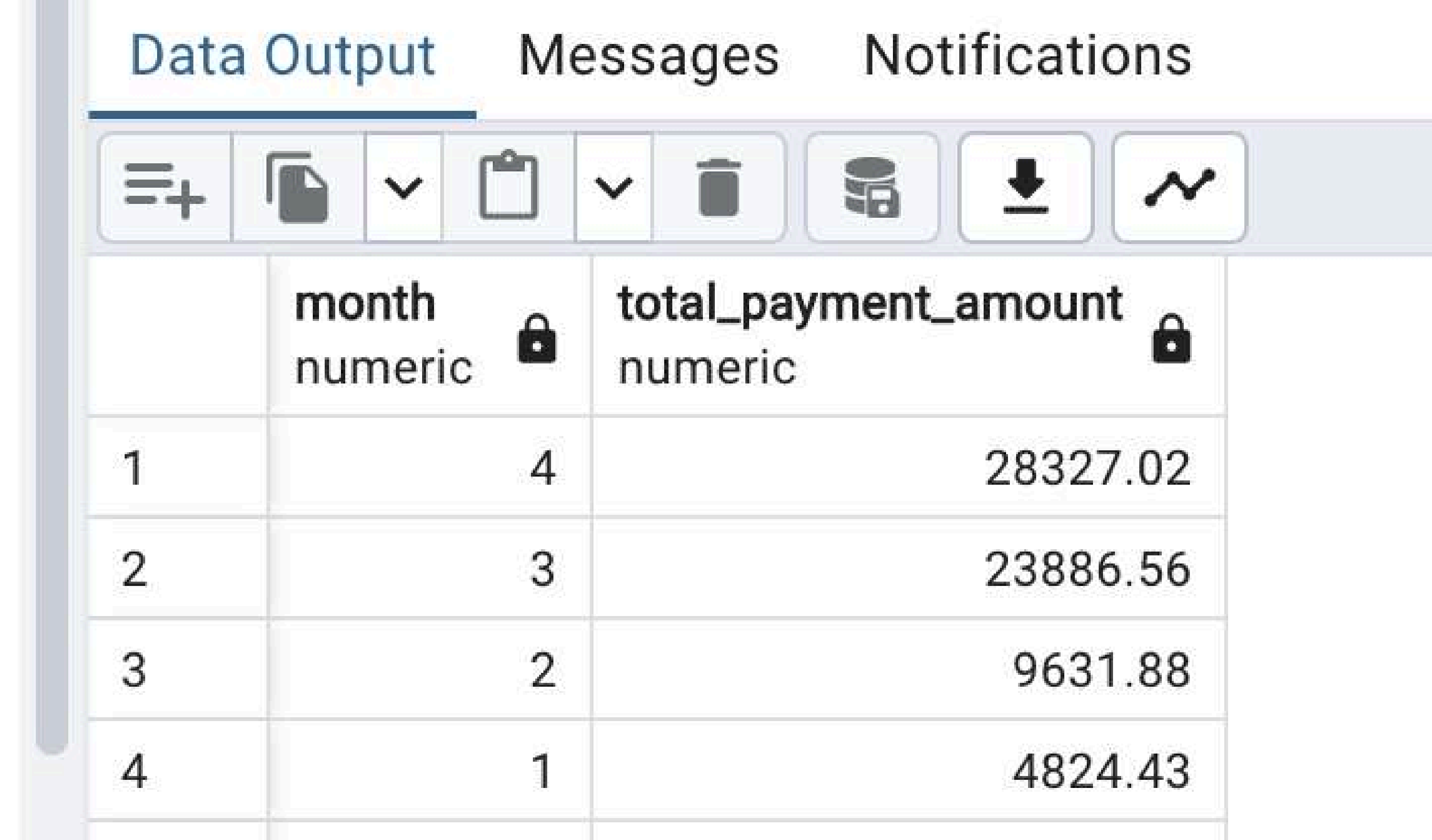


	email text	?column? text	?column? text
1	MARY.SMITH@sakilacustomer.org	M***.S***@sakilacustomer.org	***Y.S***@sakilacustomer.org
2	PATRICIA.JOHNSON@sakilacustomer.org	P***.J***@sakilacustomer.org	***A.J***@sakilacustomer.org
3	LINDA.WILLIAMS@sakilacustomer.org	L***.W***@sakilacustomer.org	***A.W***@sakilacustomer.org
4	BARBARA.JONES@sakilacustomer.org	B***.J***@sakilacustomer.org	***A.J***@sakilacustomer.org
5	ELIZABETH.BROWN@sakilacustomer.org	E***.B***@sakilacustomer.org	***H.B***@sakilacustomer.org
6	JENNIFER.DAVIS@sakilacustomer.org	J***.D***@sakilacustomer.org	***R.D***@sakilacustomer.org
7	MARIA.MILLER@sakilacustomer.org	M***.M***@sakilacustomer.o...	***A.M***@sakilacustomer.org
8	SUSAN.WILSON@sakilacustomer.org	S***.W***@sakilacustomer.org	***N.W***@sakilacustomer.org

In this challenge, you need to analyze the payments table to determine the following: First, identify the month with the highest total payment amount.

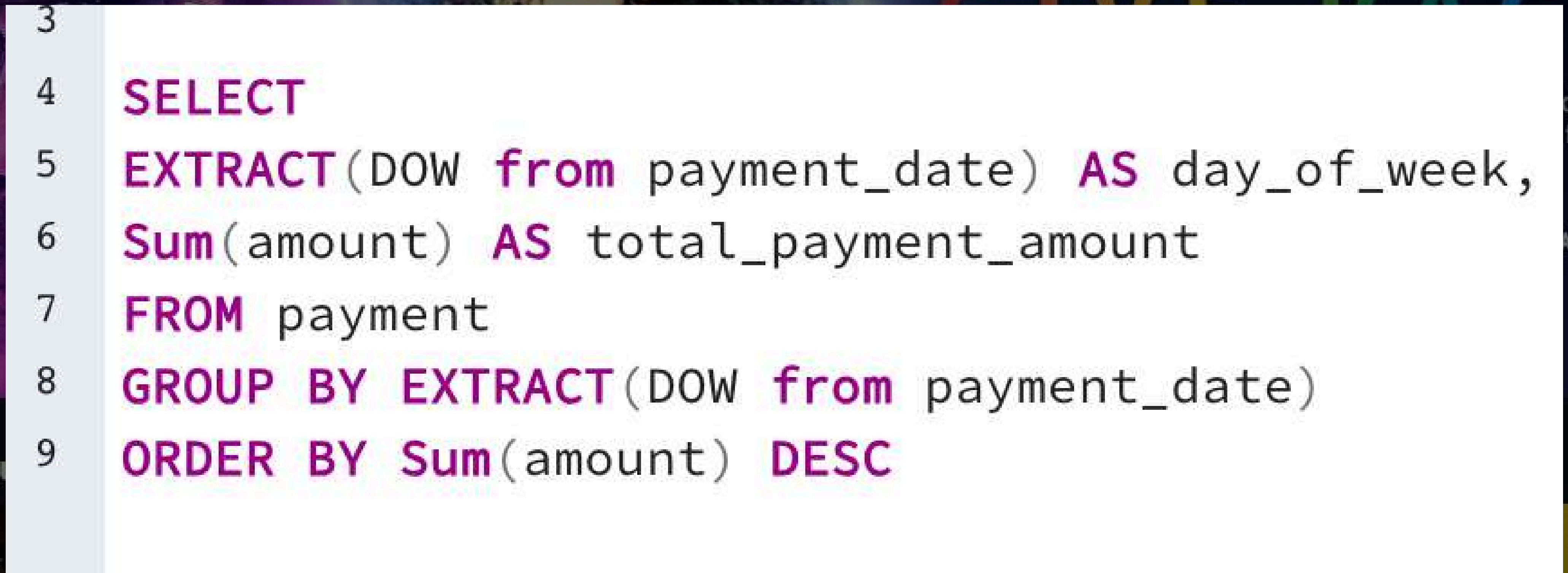


```sql
SELECT
EXTRACT(month from payment_date) AS month,
Sum(amount) AS total_payment_amount
FROM payment
GROUP BY EXTRACT(month from payment_date)
ORDER BY Sum(amount) DESC
```

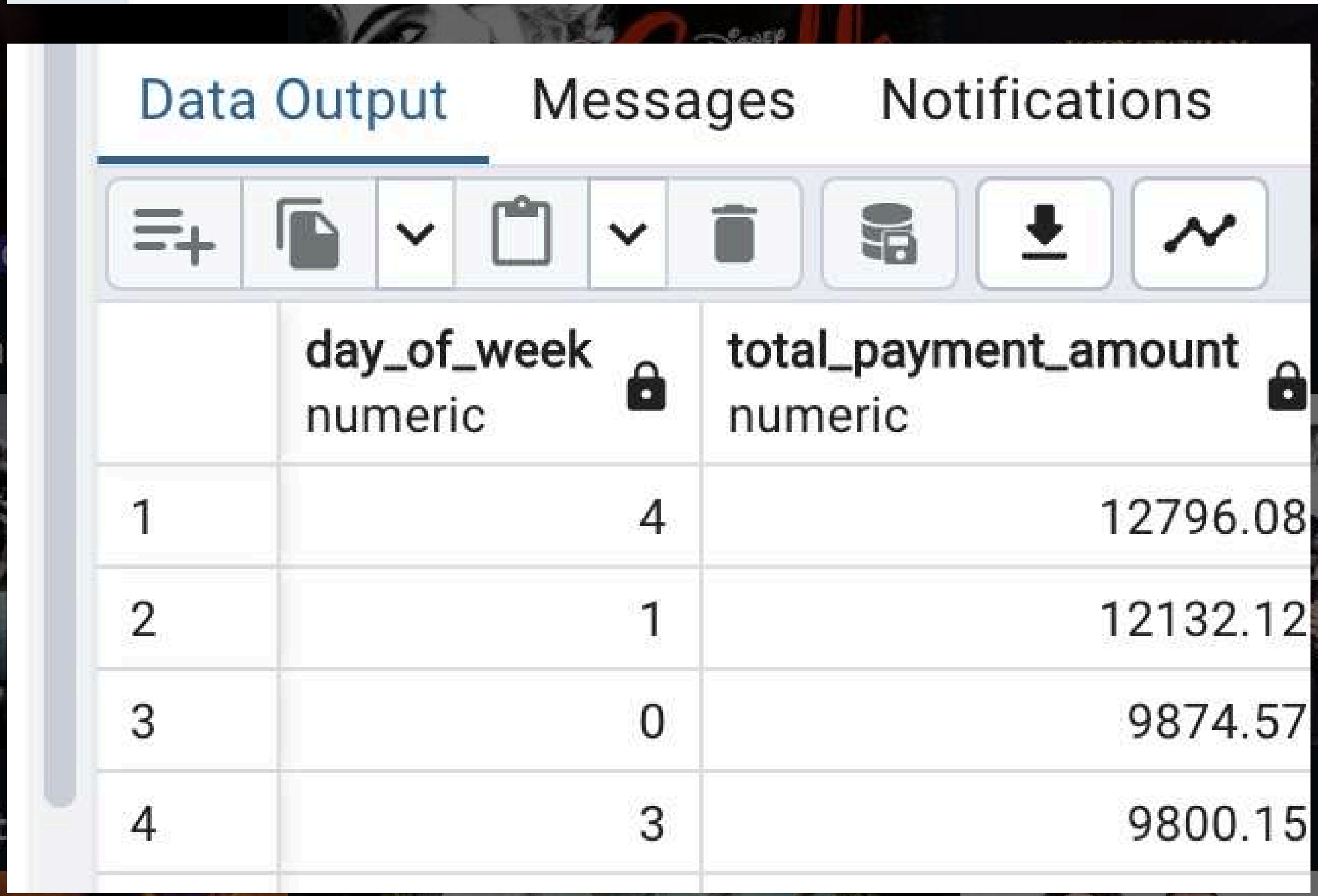


	month numeric	total_payment_amount numeric
1	4	28327.02
2	3	23886.56
3	2	9631.88
4	1	4824.43

Next, you need to identify the day of the week with the highest total payment amount, noting that zero corresponds to Sunday

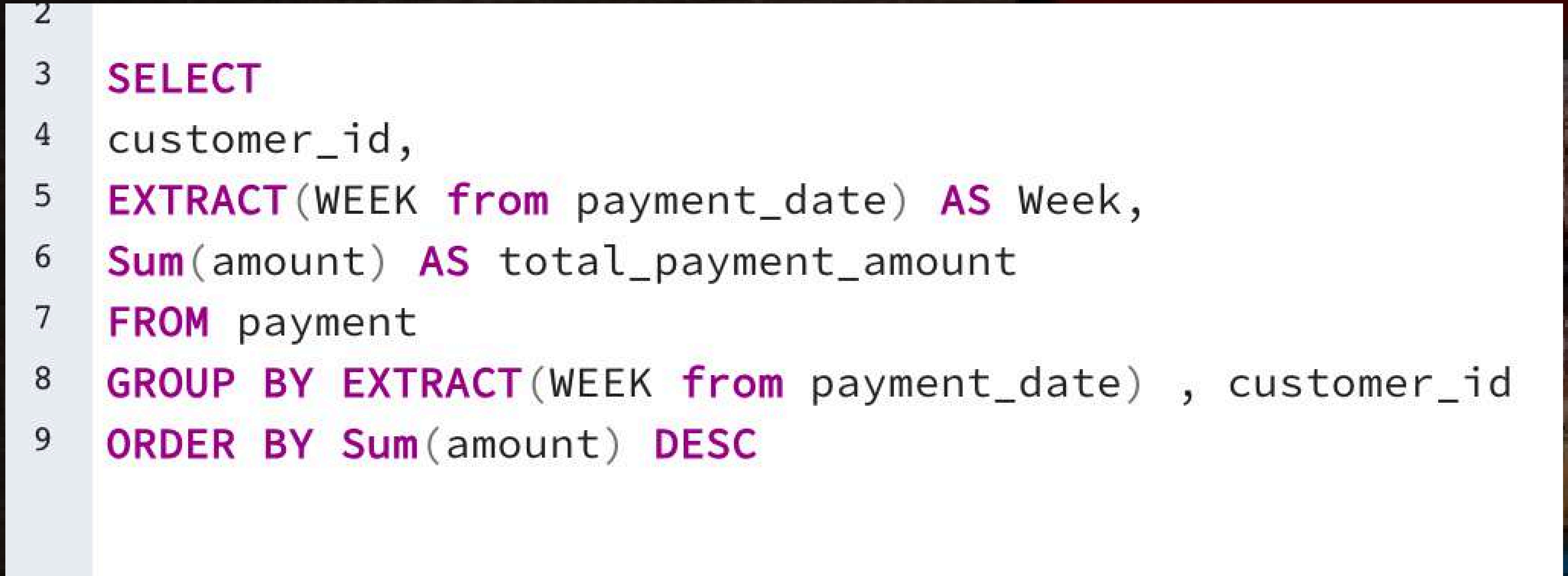


```
SELECT
EXTRACT(DOW from payment_date) AS day_of_week,
Sum(amount) AS total_payment_amount
FROM payment
GROUP BY EXTRACT(DOW from payment_date)
ORDER BY Sum(amount) DESC
```

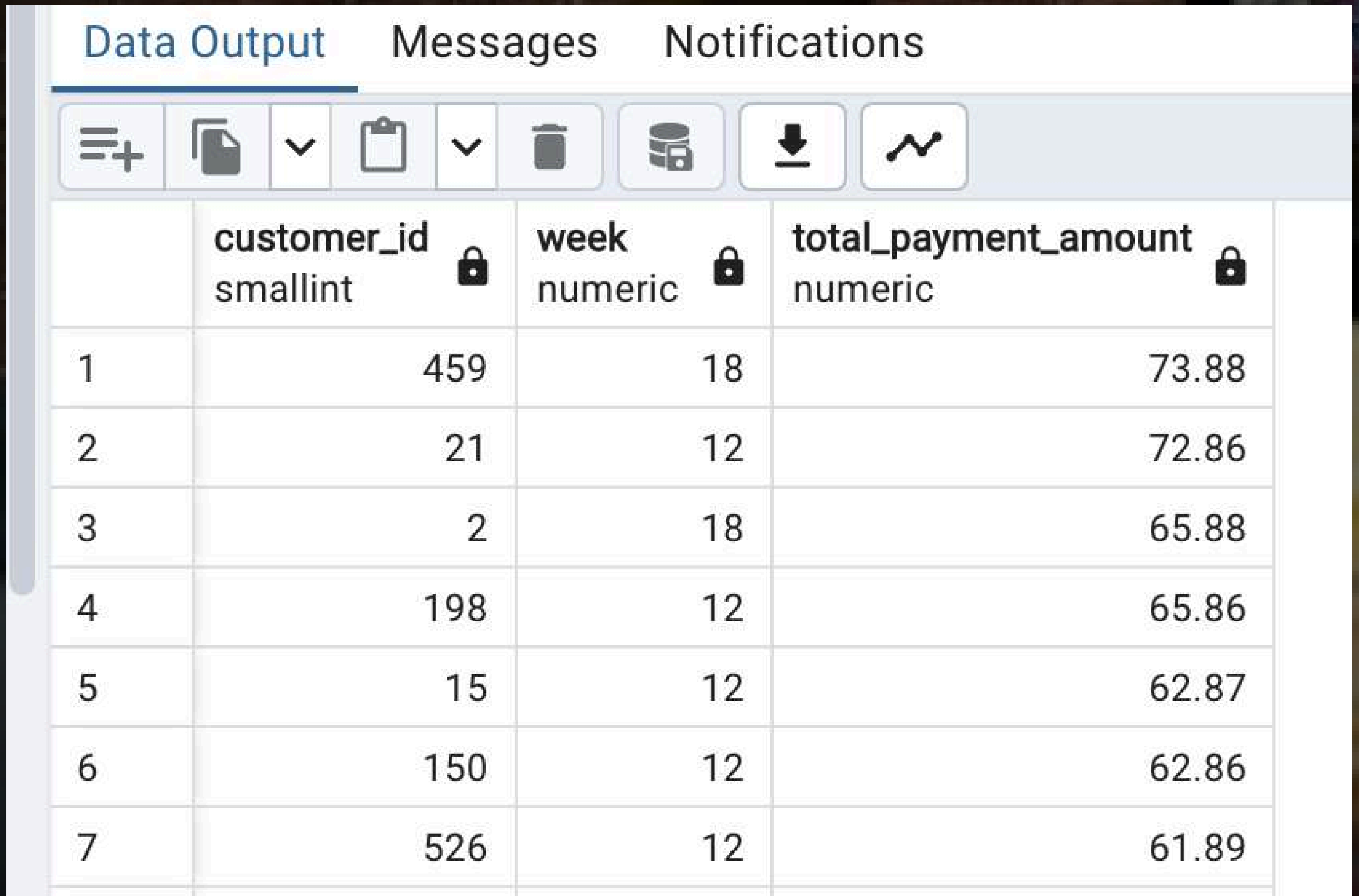


	day_of_week numeric	total_payment_amount numeric
1	4	12796.08
2	1	12132.12
3	0	9874.57
4	3	9800.15

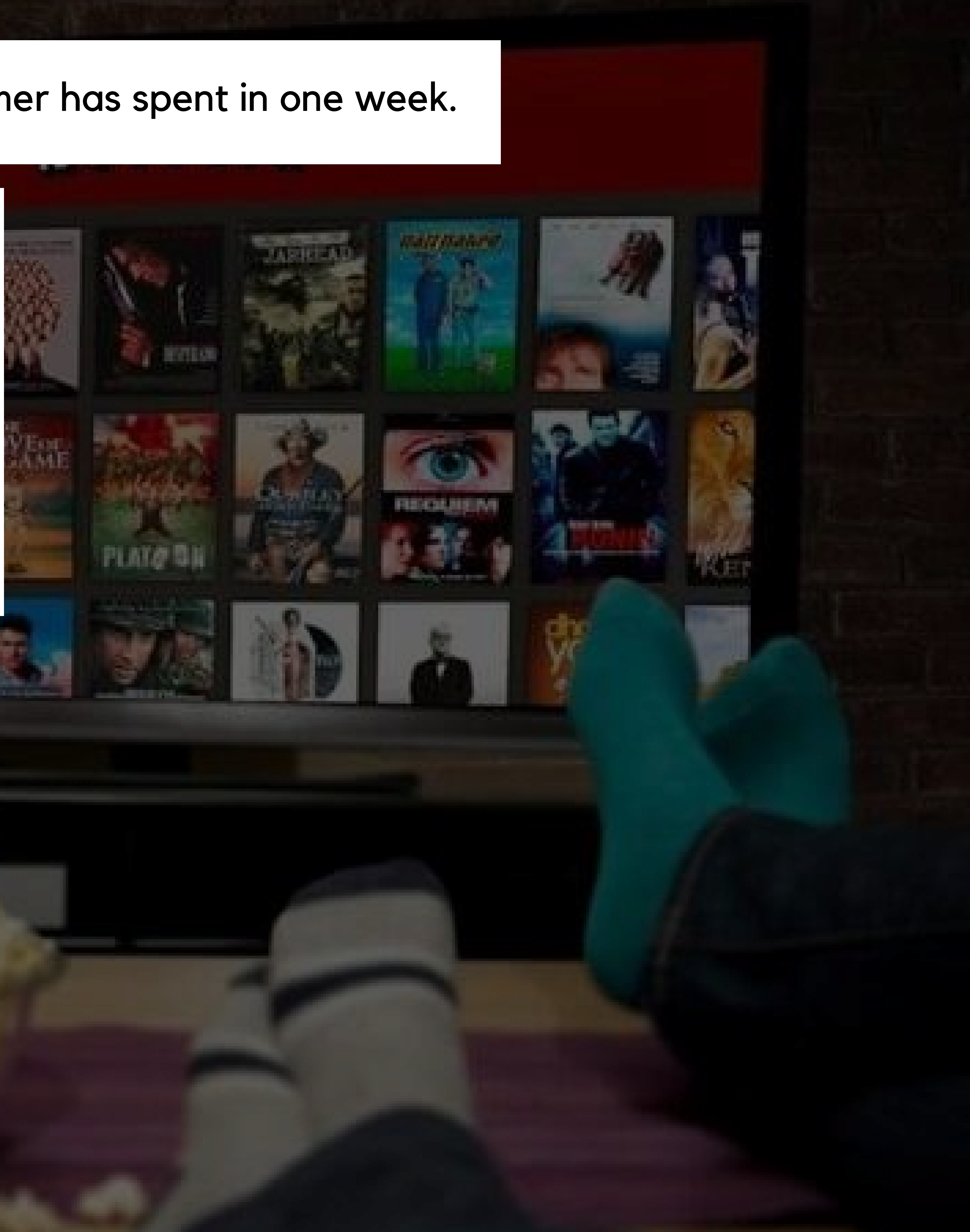
Lastly, you need to determine the highest amount a single customer has spent in one week.



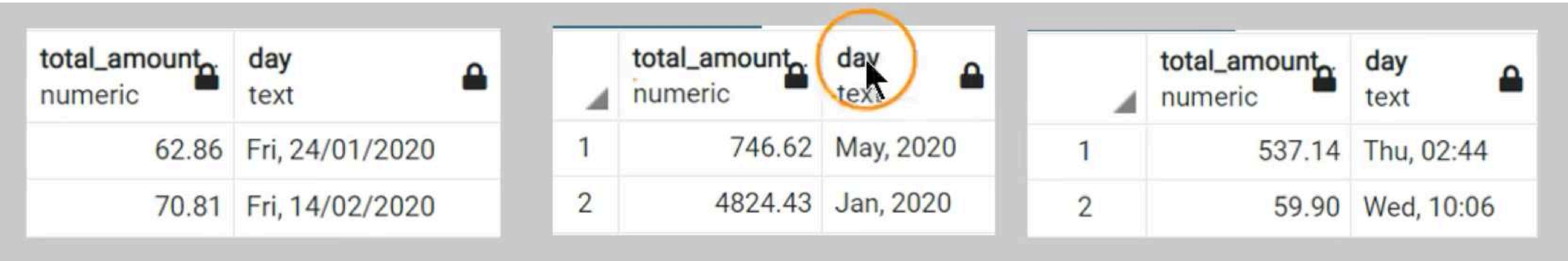
```sql
SELECT
customer_id,
EXTRACT(WEEK from payment_date) AS Week,
Sum(amount) AS total_payment_amount
FROM payment
GROUP BY EXTRACT(WEEK from payment_date) , customer_id
ORDER BY Sum(amount) DESC
```



	customer_id smallint	week numeric	total_payment_amount numeric
1	459	18	73.88
2	21	12	72.86
3	2	18	65.88
4	198	12	65.86
5	15	12	62.87
6	150	12	62.86
7	526	12	61.89



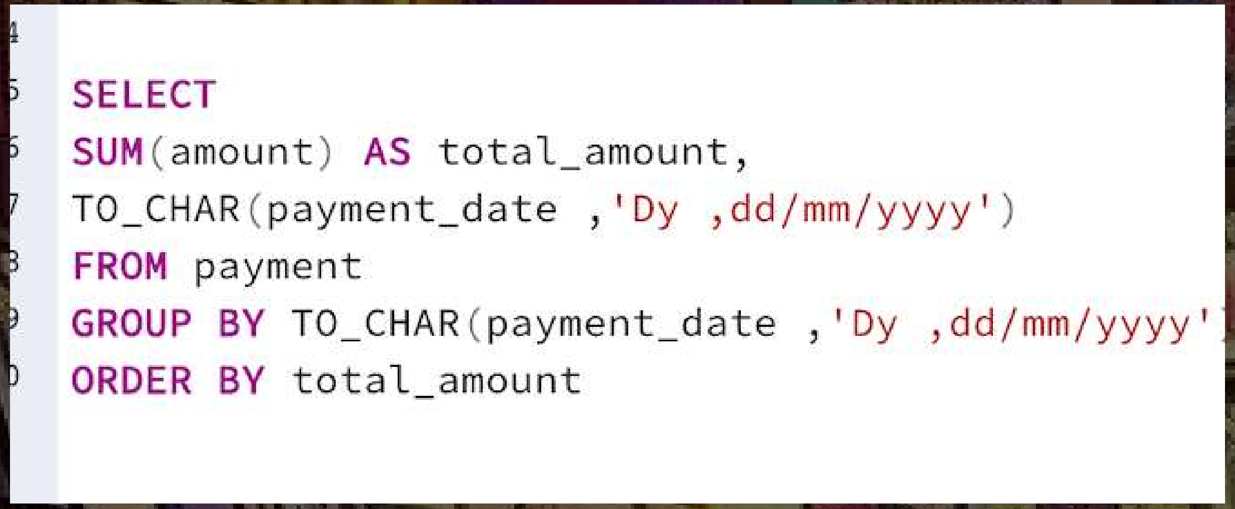
You need to total the payment amounts and organize them into specific formats as outlined by the company's guidelines.



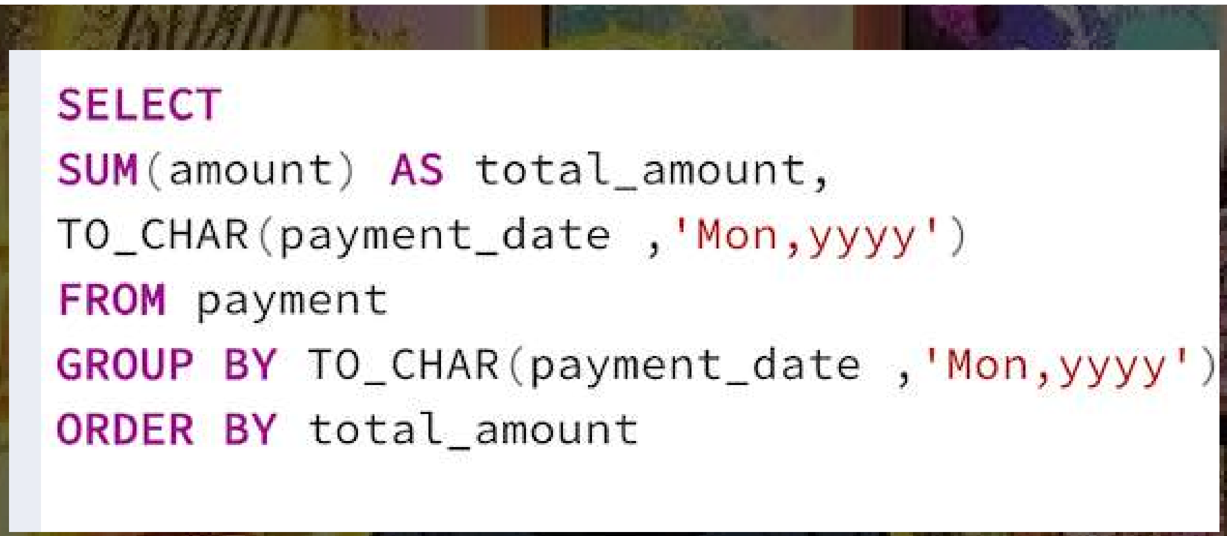
total_amount numeric	day text
62.86	Fri, 24/01/2020
70.81	Fri, 14/02/2020

	total_amount numeric	day text
1	746.62	May, 2020
2	4824.43	Jan, 2020

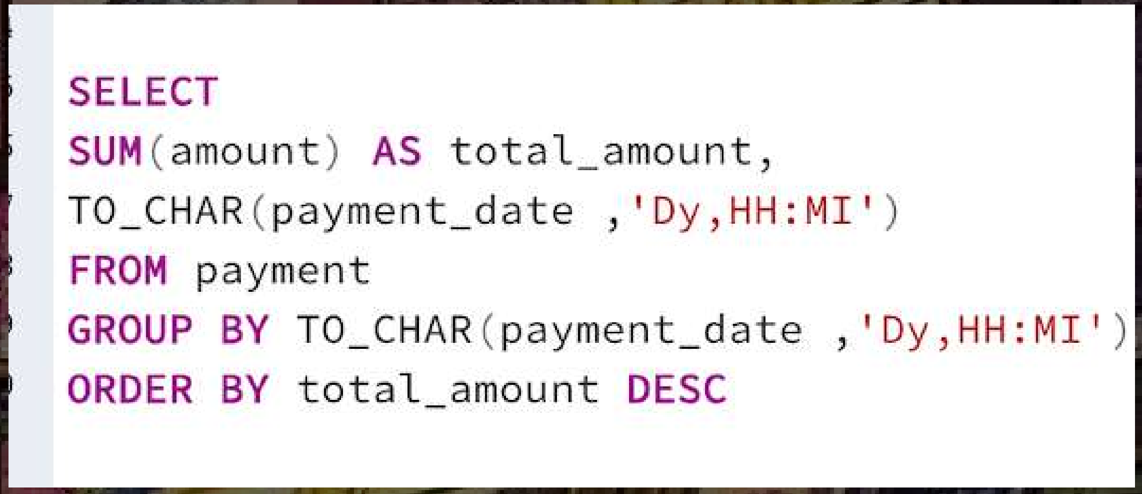
	total_amount numeric	day text
1	537.14	Thu, 02:44
2	59.90	Wed, 10:06



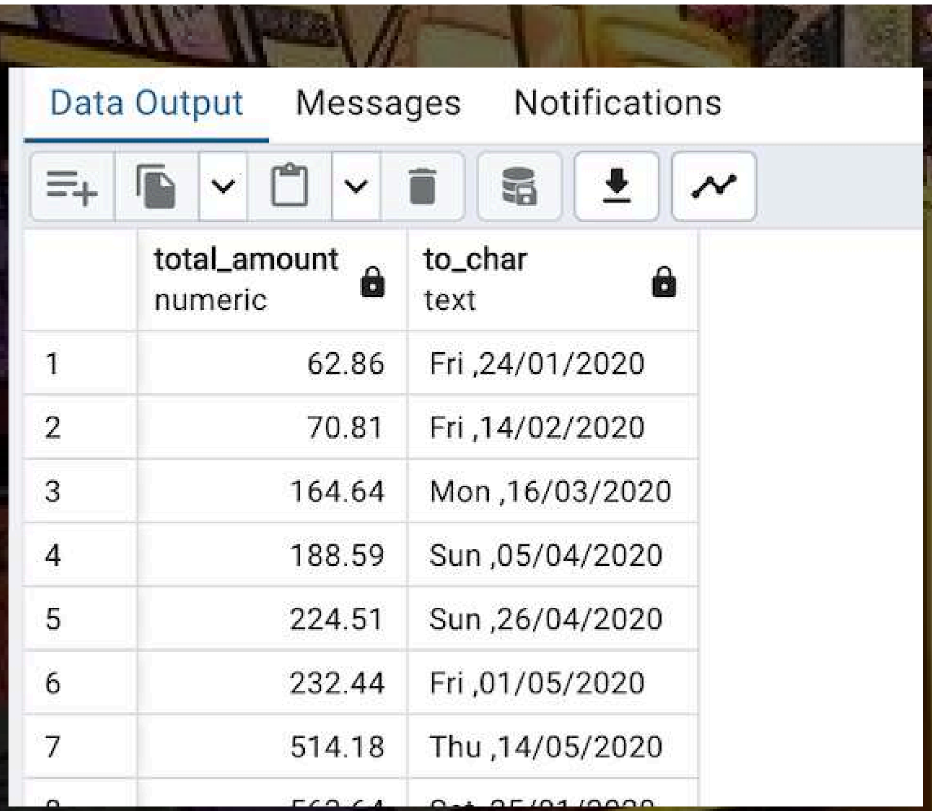
```
SELECT
SUM(amount) AS total_amount,
TO_CHAR(payment_date ,'Dy ,dd/mm/yyyy')
FROM payment
GROUP BY TO_CHAR(payment_date ,'Dy ,dd/mm/yyyy'
ORDER BY total_amount
```



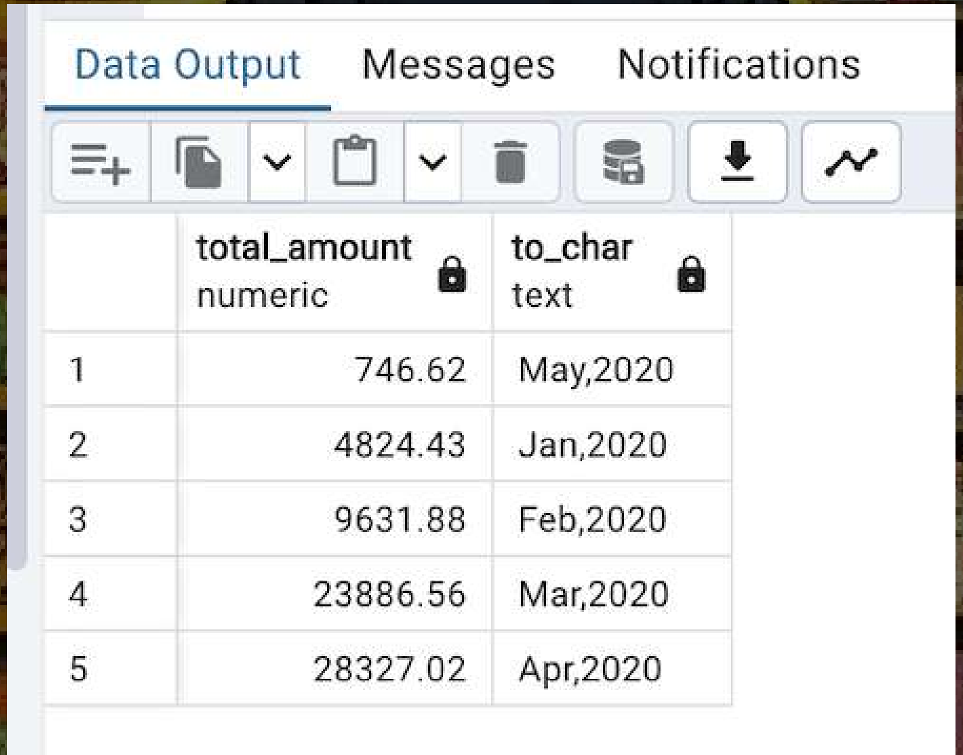
```
SELECT
SUM(amount) AS total_amount,
TO_CHAR(payment_date ,'Mon,yyyy')
FROM payment
GROUP BY TO_CHAR(payment_date ,'Mon,yyyy')
ORDER BY total_amount
```



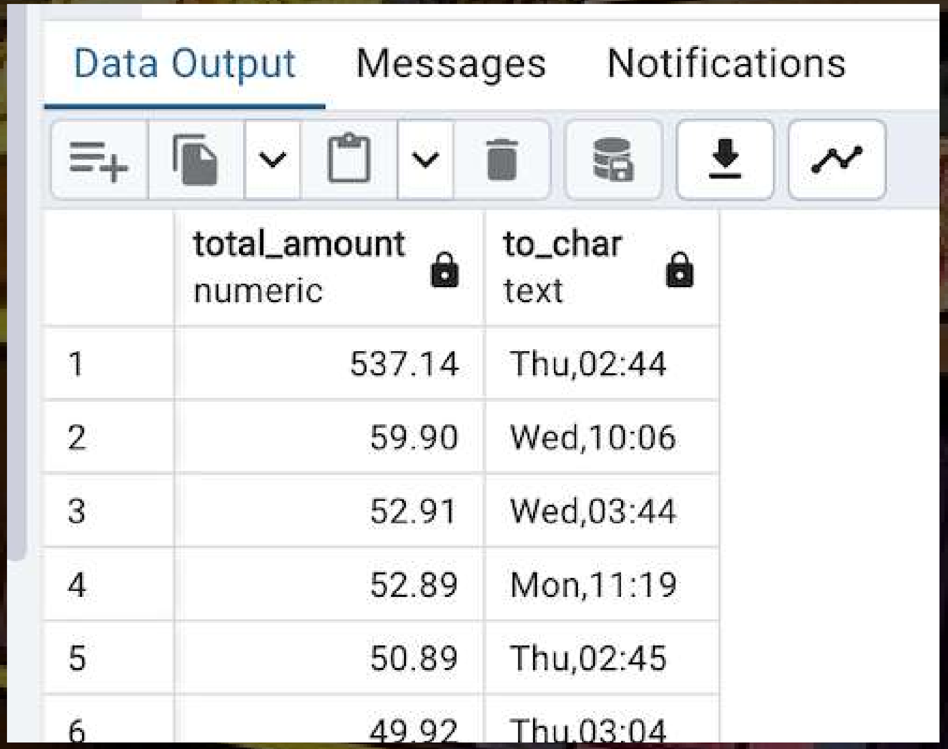
```
SELECT
SUM(amount) AS total_amount,
TO_CHAR(payment_date ,'Dy,HH:MI')
FROM payment
GROUP BY TO_CHAR(payment_date ,'Dy,HH:MI')
ORDER BY total_amount DESC
```



	total_amount numeric	to_char text
1	62.86	Fri ,24/01/2020
2	70.81	Fri ,14/02/2020
3	164.64	Mon ,16/03/2020
4	188.59	Sun ,05/04/2020
5	224.51	Sun ,26/04/2020
6	232.44	Fri ,01/05/2020
7	514.18	Thu ,14/05/2020
8	563.64	Sat ,25/01/2020

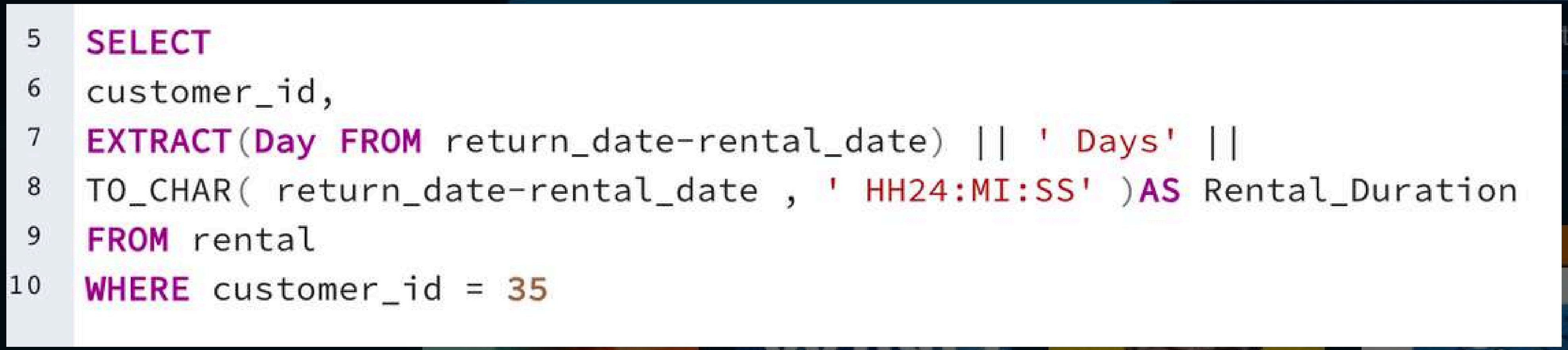


	total_amount numeric	to_char text
1	746.62	May,2020
2	4824.43	Jan,2020
3	9631.88	Feb,2020
4	23886.56	Mar,2020
5	28327.02	Apr,2020

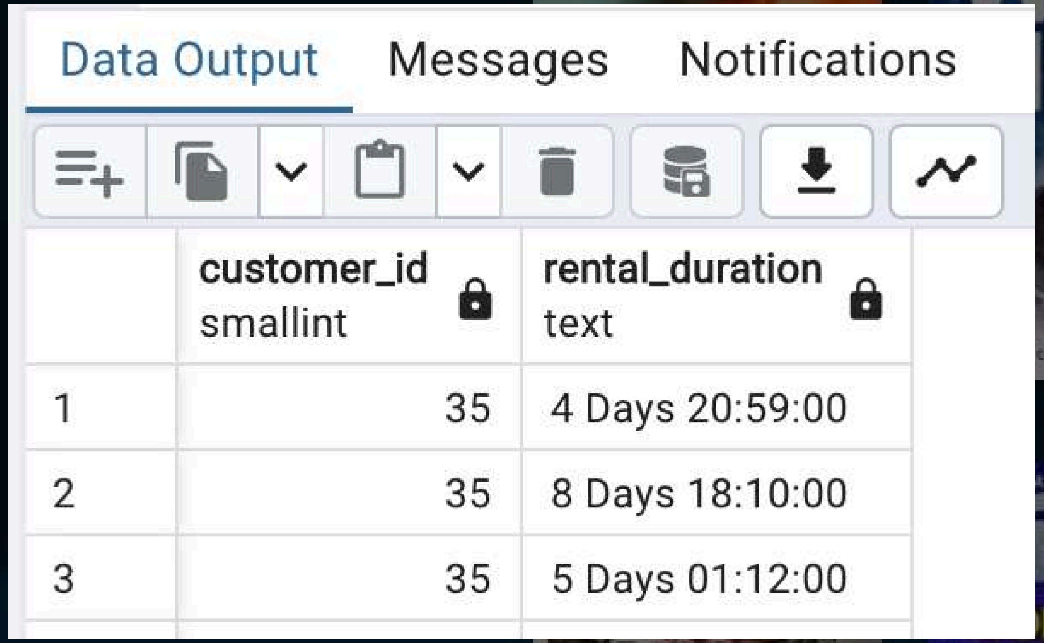


	total_amount numeric	to_char text
1	537.14	Thu,02:44
2	59.90	Wed,10:06
3	52.91	Wed,03:44
4	52.89	Mon,11:19
5	50.89	Thu,02:45
6	49.92	Thu,03:04

For this task, the support team is requesting you to compile a list of all rental durations for the customer with ID 35.

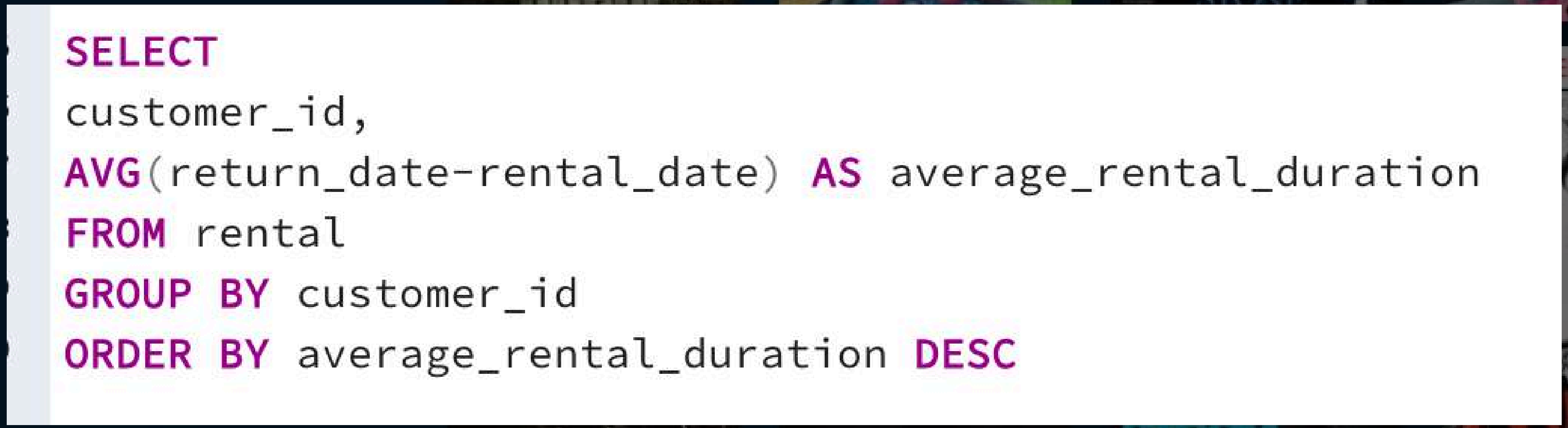


```sql
SELECT
customer_id,
EXTRACT(Day FROM return_date-rental_date) || ' Days' ||
TO_CHAR( return_date-rental_date , ' HH24:MI:SS' )AS Rental_Duration
FROM rental
WHERE customer_id = 35
```

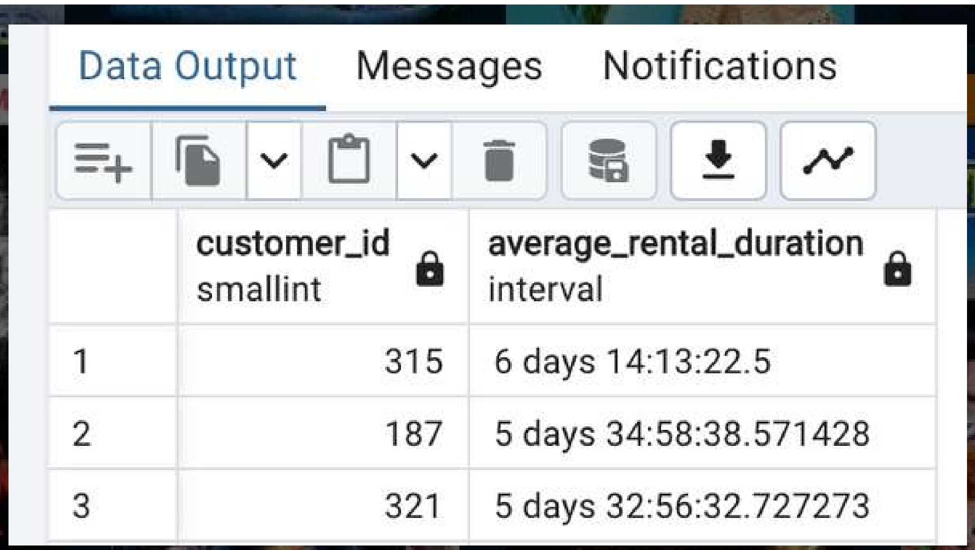


	customer_id smallint	rental_duration text
1	35	4 Days 20:59:00
2	35	8 Days 18:10:00
3	35	5 Days 01:12:00

After that, the support team is also asking you to identify the customer with the longest average rental duration. This information is needed to help them address some issues.

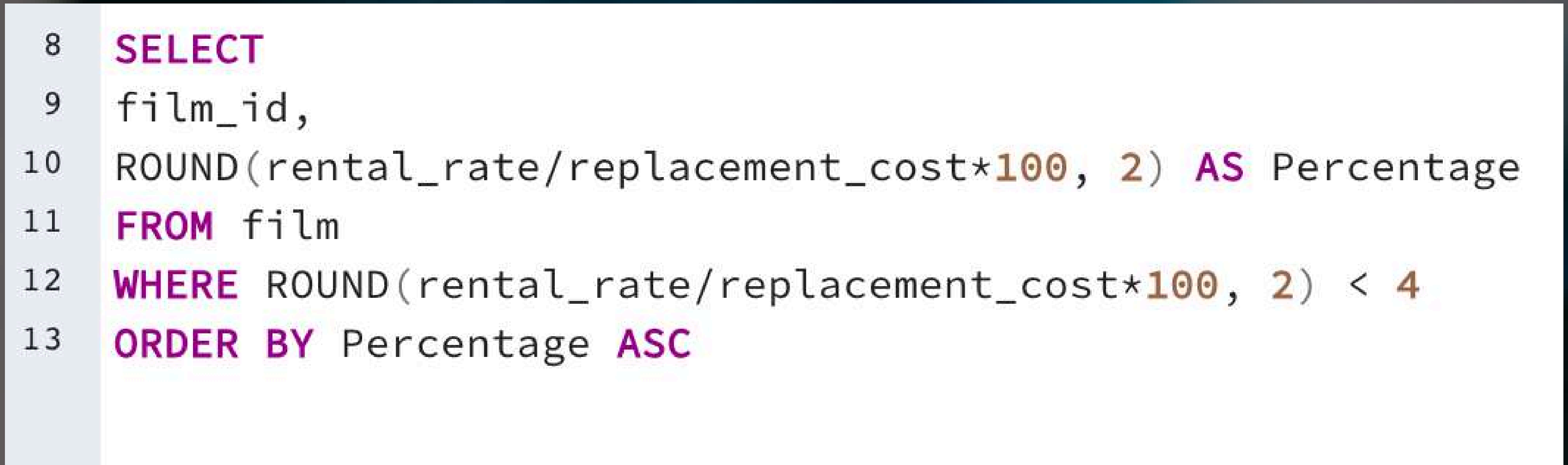


```sql
SELECT
customer_id,
AVG(return_date-rental_date) AS average_rental_duration
FROM rental
GROUP BY customer_id
ORDER BY average_rental_duration DESC
```

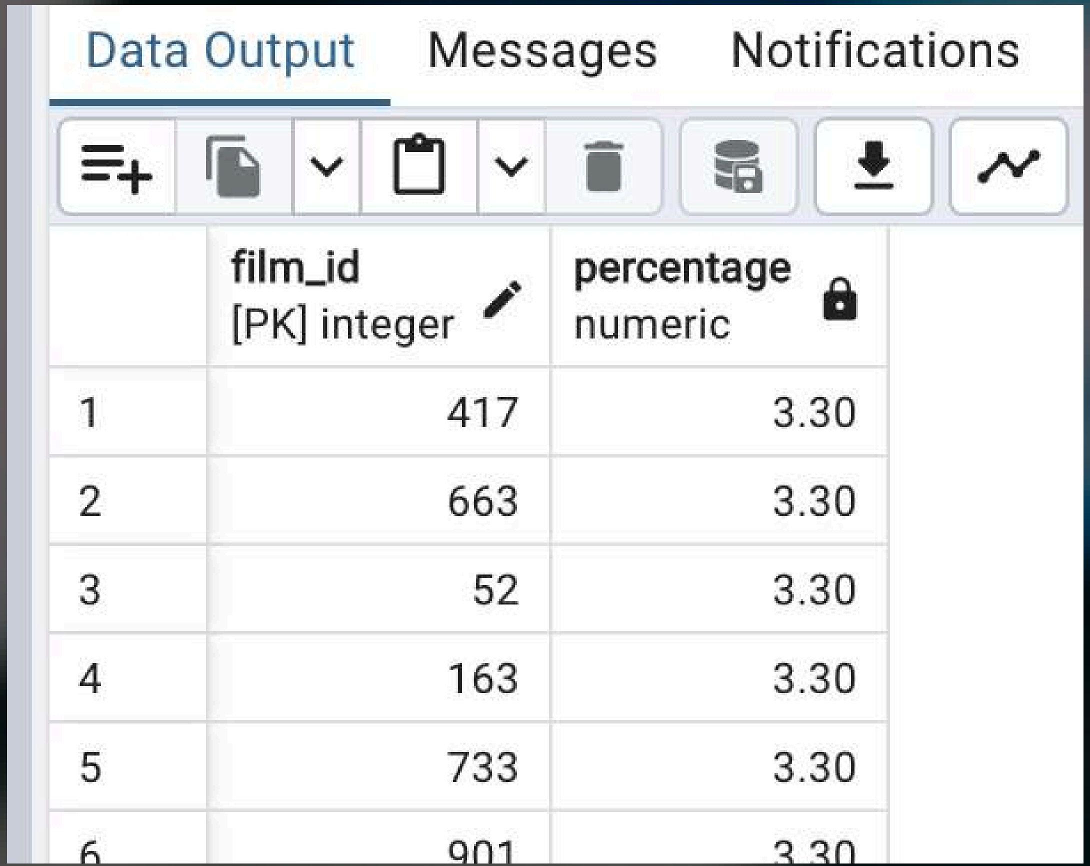


	customer_id smallint	average_rental_duration interval
1	315	6 days 14:13:22.5
2	187	5 days 34:58:38.571428
3	321	5 days 32:56:32.727273

In this challenge, your manager wants to raise the prices for films that are very costly to replace and currently have a rental rate that is low compared to their replacement costs. To address this, you should compile a list of films where the rental rate is less than 4% of the replacement cost. Include the percentage value of the rental rate in relation to the replacement cost for each film, rounded to two decimal places. The result should look like this: 3.54, meaning 3.54%.

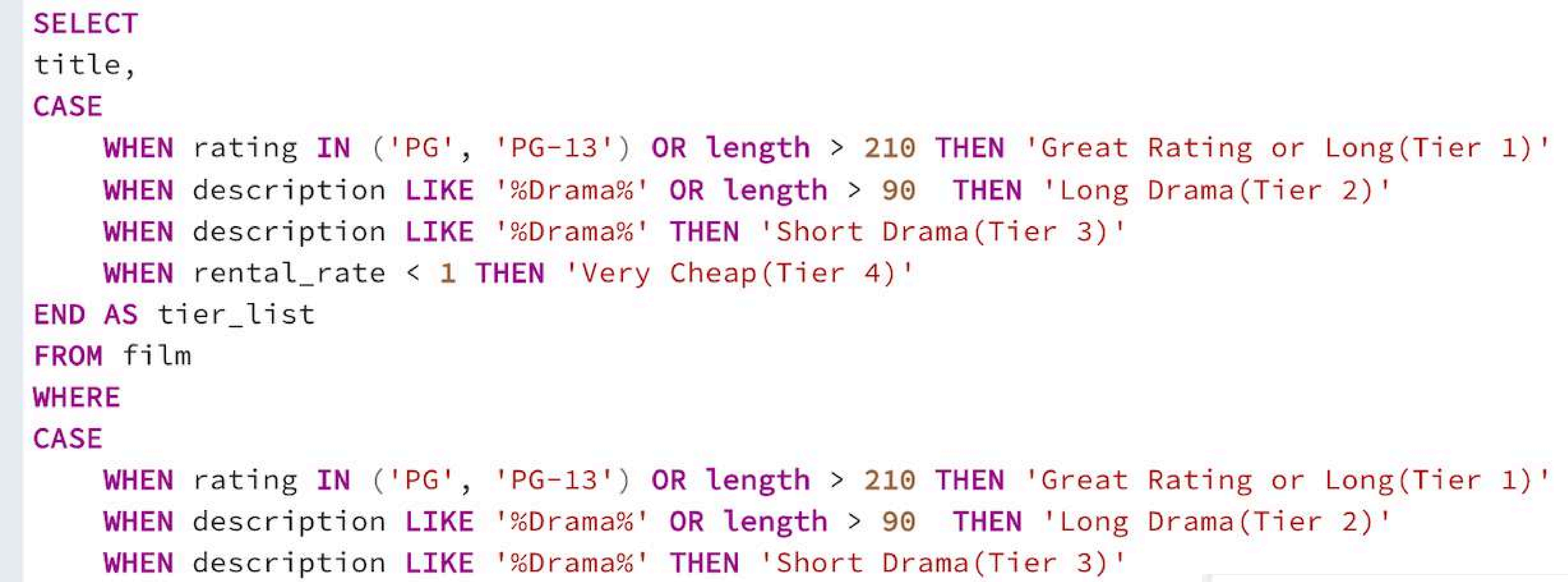


```sql
SELECT
film_id,
ROUND(rental_rate/replacement_cost*100, 2) AS Percentage
FROM film
WHERE ROUND(rental_rate/replacement_cost*100, 2) < 4
ORDER BY Percentage ASC
```

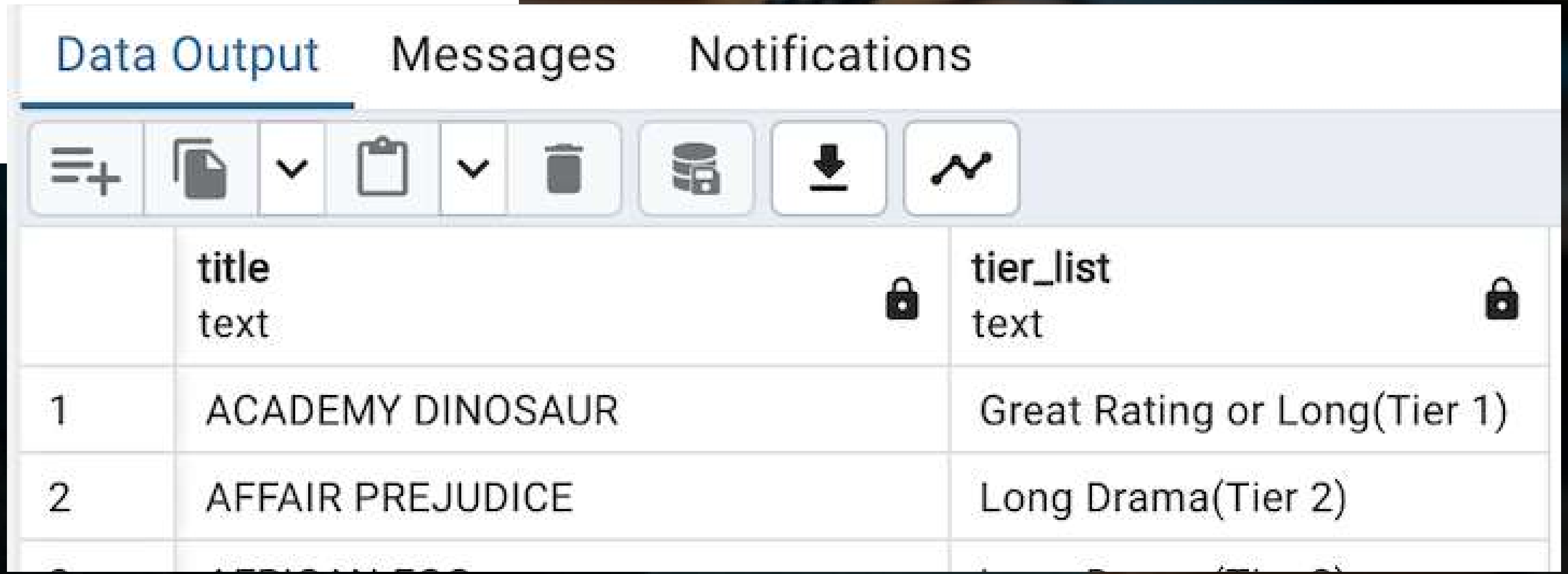


	film_id [PK] integer	percentage numeric
1	417	3.30
2	663	3.30
3	52	3.30
4	163	3.30
5	733	3.30
6	901	3.30

In this task, we need to create a tier list of movies. Tier One includes films rated PG or PG-13, or over 210 minutes long, labeled "Great Rating or Long." Tier Two includes films with "drama" in the description and over 90 minutes long, labeled "Long Drama." Tier Three covers films with "drama" and 90 minutes or less, labeled "Short Drama." Tier Four includes movies with a rental rate under $1. The challenge is to filter the list to include only movies in these tiers, excluding those not assigned to any tier. How would you implement this filtering process?



```
SELECT
title,
CASE
    WHEN rating IN ('PG', 'PG-13') OR length > 210 THEN 'Great Rating or Long(Tier 1)'
    WHEN description LIKE '%Drama%' OR length > 90  THEN 'Long Drama(Tier 2)'
    WHEN description LIKE '%Drama%' THEN 'Short Drama(Tier 3)'
    WHEN rental_rate < 1 THEN 'Very Cheap(Tier 4)'
END AS tier_list
FROM film
WHERE
CASE
    WHEN rating IN ('PG', 'PG-13') OR length > 210 THEN 'Great Rating or Long(Tier 1)'
    WHEN description LIKE '%Drama%' OR length > 90  THEN 'Long Drama(Tier 2)'
    WHEN description LIKE '%Drama%' THEN 'Short Drama(Tier 3)'
    WHEN rental_rate < 1 THEN 'Very Cheap(Tier 4)'
END IS NOT null
```



	title text	tier_list text
1	ACADEMY DINOSAUR	Great Rating or Long(Tier 1)
2	AFFAIR PREJUDICE	Long Drama(Tier 2)

Project Summary

In this project, we addressed a variety of queries to fulfill the requirements of different departments within the company. Here are some tasks we handled during the project:

- Customer List, Sorted Customers, Payment Prices
- Film Count, Latest Rental Date, Most Recent Rental Date
- Unreturned Rentals & Low Payments, Specific Customer Payments, Faulty Transactions, Complaint Analysis, Customer Name Criteria
- Film Costs, Employee Payments, Employee Sales, High Revenue Days

We used basic SQL queries, such as `SELECT`, `ORDER BY`, `LIMIT`, `DISTINCT`, and filtering functions like `WHERE` and `HAVING`. Aggregation functions like `SUM`, `AVG`, `MAX`, and `MIN`, along with `GROUP BY`, and functions like `LEFT`, `RIGHT`, `SUBSTRING`, `POSITION`, `LOWER`, `UPPER`, `LENGTH`, `EXTRACT`, `TO_CHAR`, `CASE WHEN`, `ROUND`, and basic SQL mathematical functions were utilized to generate the necessary insights and support effective data management and analysis.

This approach ensured we delivered accurate and actionable data to support the company’s diverse operational needs.

Home
Store
Live TV
Categories
My Stuff
Rent
EARLY ACCESS
CINEMA-LIKE EXPERIENCE AT HOME.
RENT AND WATCH THE LATEST MOVIES
KILLERS OF THE FLOWER MOON
OPPENHEIMER
Rent

Thank you..