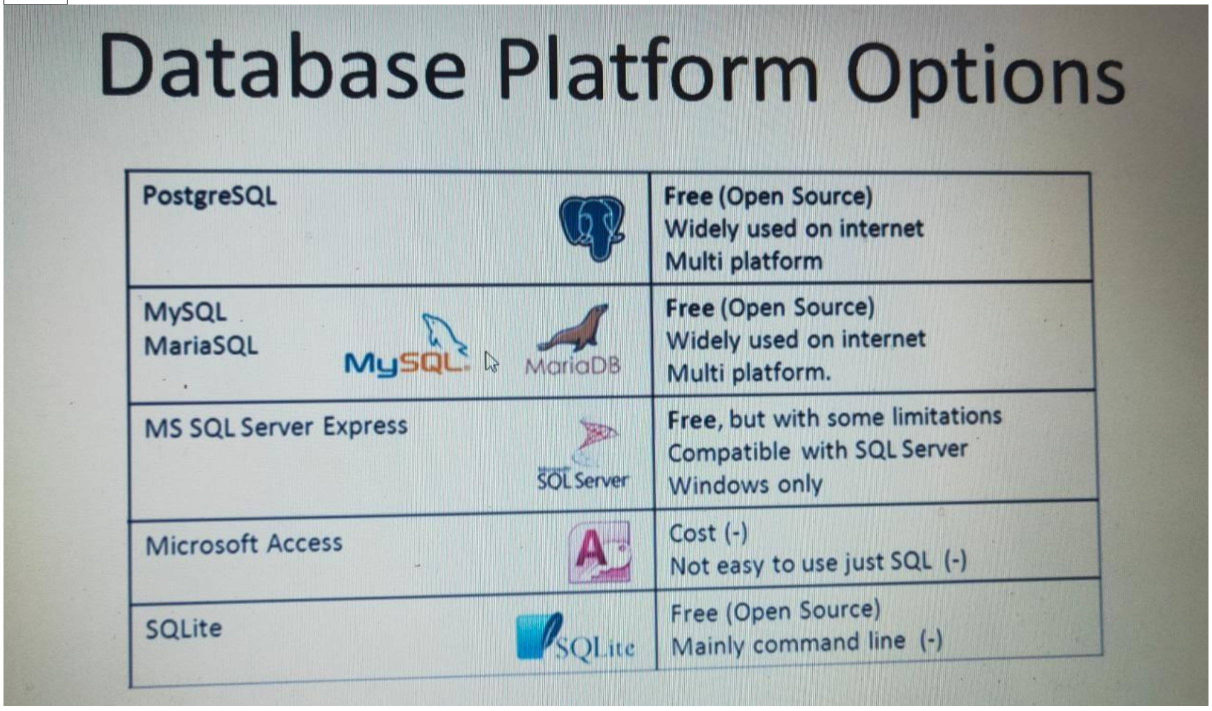
POSTGRE-SQL

What are Database:
DB are systems that allow users to store and organize data.
They are useful when dealing with large amounts of data

Users:
Marketing,business,sales,data science,software enginersand web developers



Database Platform Options

PostgreSQL	Free (Open Source) Widely used on internet Multi platform
MySQL MariaSQL	Free (Open Source) Widely used on internet Multi platform.
MS SQL Server Express	Free, but with some limitations Compatible with SQL Server Windows only
Microsoft Access	Cost (-) Not easy to use just SQL (-)
SQLite	Free (Open Source) Mainly command line (-)

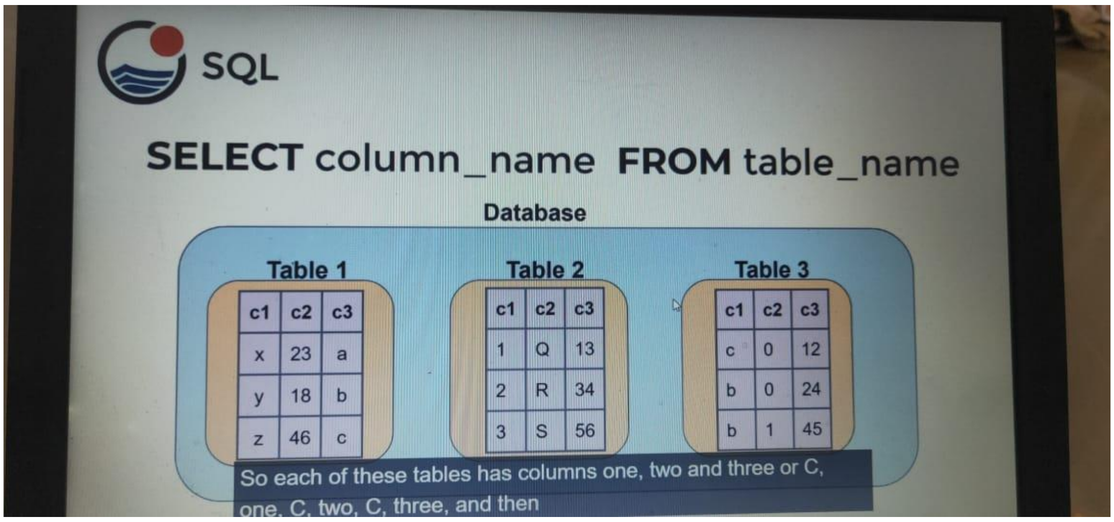
SQL :
It is the programming language used to communicate with our DB

SELECT :
It is the most common statement used, and it allows us to retrieve information from a table

EAMPLE; SELECT column_name FROM table_name ;

{caps SELECT and FROM is just for our refferences to avoid confusion in the query so we can use lower case select }

; - indicating the end of the quey so without ; it would be error



SQL

SELECT column_name FROM table_name

Database

Table 1

c1	c2	c3
x	23	a
y	18	b
z	46	c

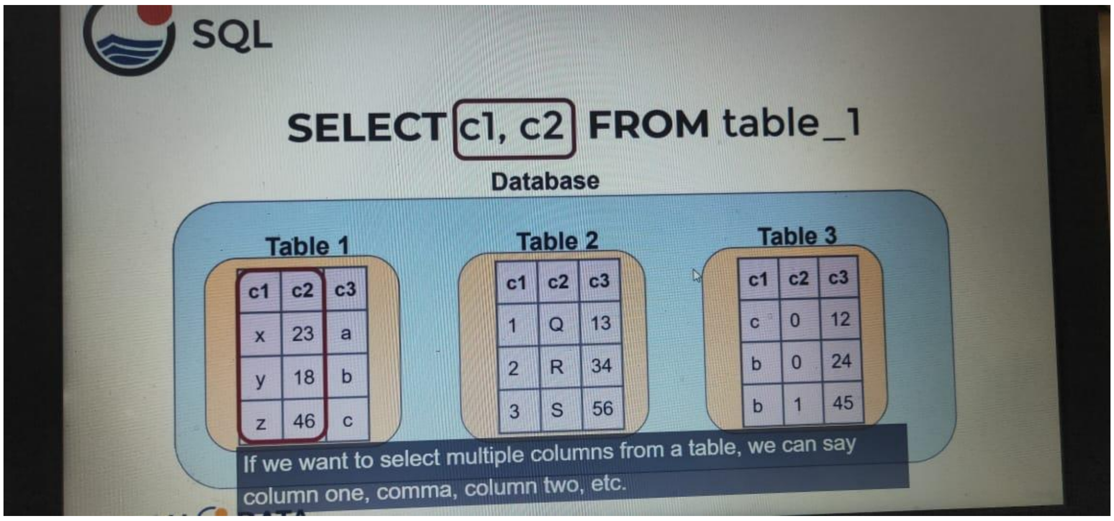
Table 2

c1	c2	c3
1	Q	13
2	R	34
3	S	56

Table 3

c1	c2	c3
c	0	12
b	0	24
b	1	45

So each of these tables has columns one, two and three or C, one, C, two, C, three, and then



SQL

SELECT $\boxed{\text{c1, c2}}$ FROM table_1

Database

Table 1

c1	c2	c3
x	23	a
y	18	b
z	46	c

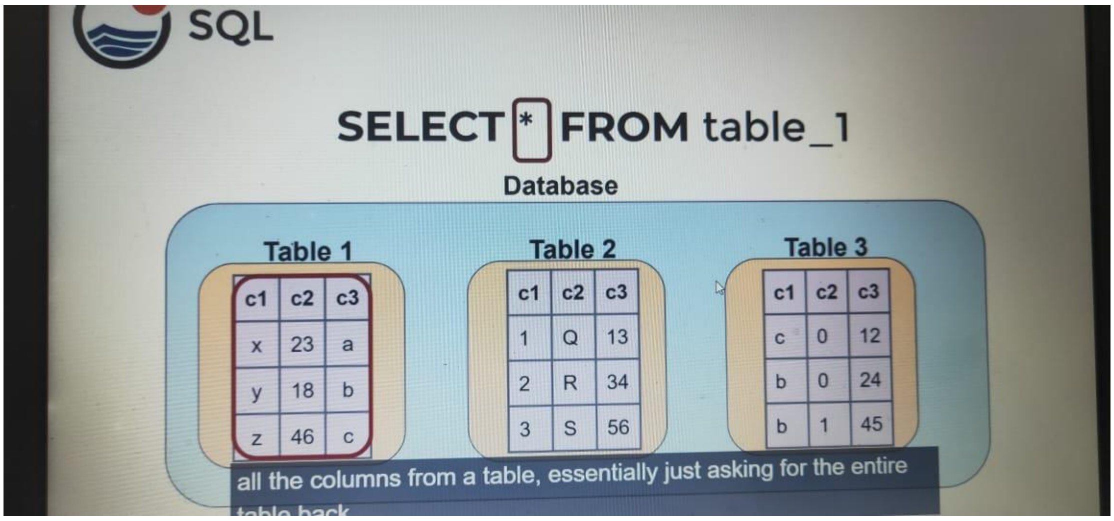
Table 2

c1	c2	c3
1	Q	13
2	R	34
3	S	56

Table 3

c1	c2	c3
c	0	12
b	0	24
b	1	45

If we want to select multiple columns from a table, we can say column one, comma, column two, etc.



SELECT $\boxed{*}$ FROM table_1

Database

Table 1

c1	c2	c3
x	23	a
y	18	b
z	46	c

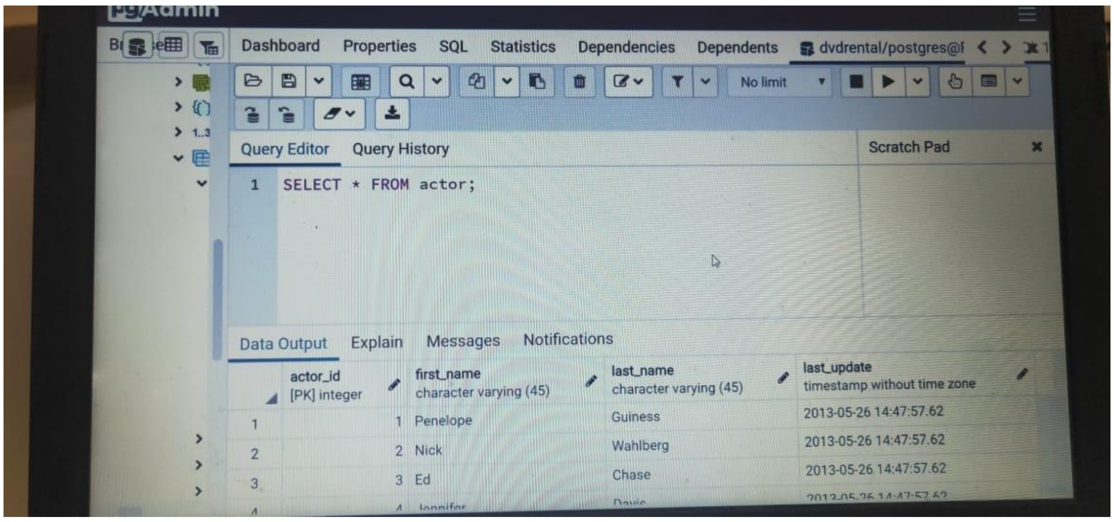
Table 2

c1	c2	c3
1	Q	13
2	R	34
3	S	56

Table 3

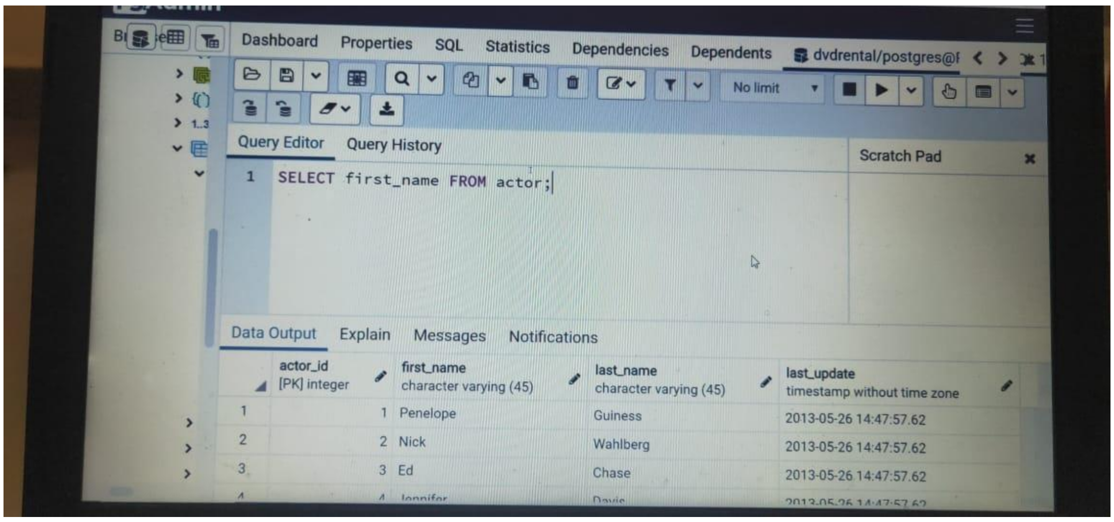
c1	c2	c3
c	0	12
b	0	24
b	1	45

all the columns from a table, essentially just asking for the entire table back



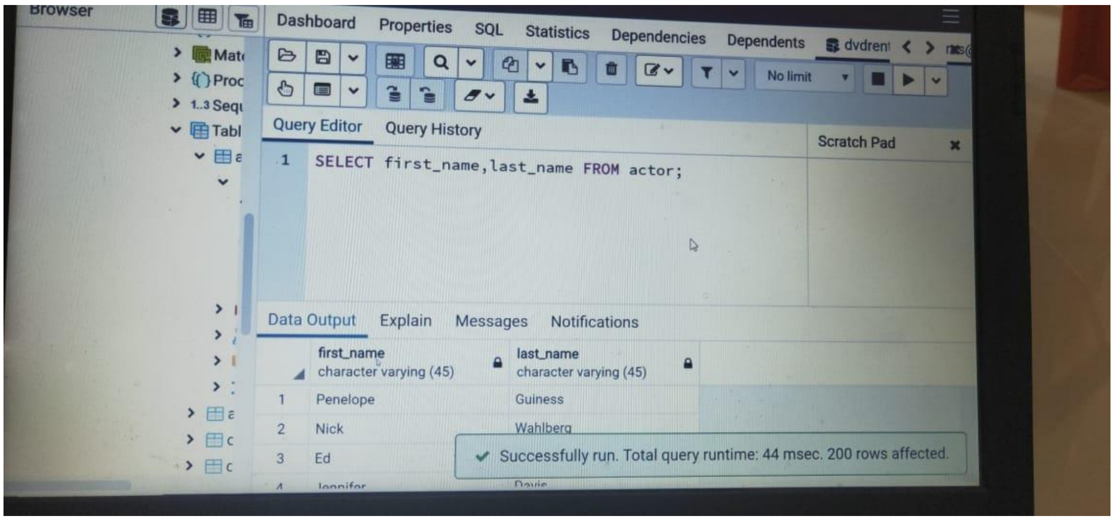
```sql
SELECT * FROM actor;
```

```
   actor_id       first_name              last_name               last_update
   [PK] integer   character varying (45)  character varying (45)  timestamp without time zone
1             1   Penelope                Guiness                 2013-05-26 14:47:57.62
2             2   Nick                    Wahlberg                2013-05-26 14:47:57.62
3             3   Ed                      Chase                   2013-05-26 14:47:57.62
4             4   Jennifer                Davis                   2013-05-26 14:47:57.62
```



```sql
SELECT first_name FROM actor;
```

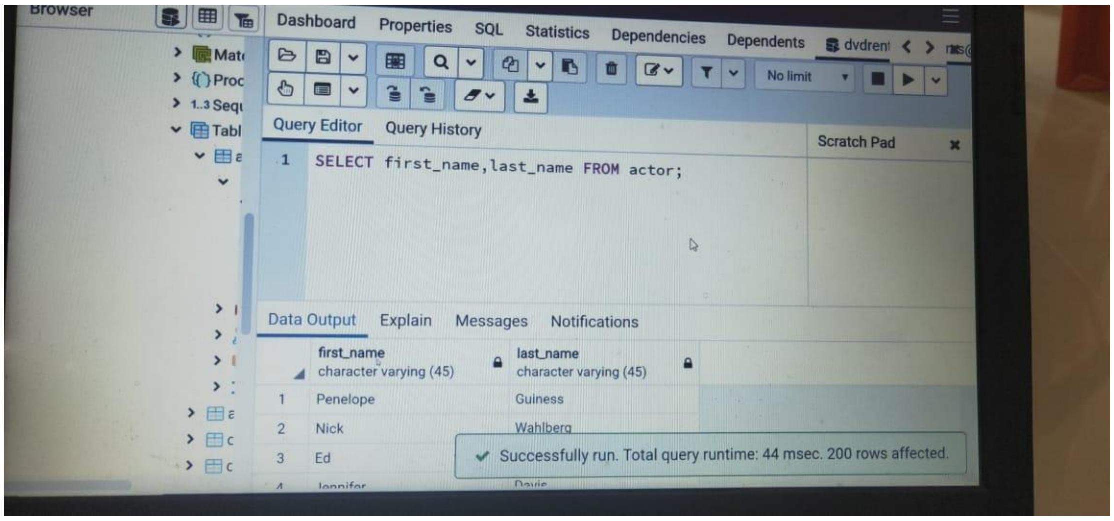
```
   actor_id       first_name              last_name               last_update
   [PK] integer   character varying (45)  character varying (45)  timestamp without time zone
1             1   Penelope                Guiness                 2013-05-26 14:47:57.62
2             2   Nick                    Wahlberg                2013-05-26 14:47:57.62
3             3   Ed                      Chase                   2013-05-26 14:47:57.62
4             4   Jennifer                Davis                   2013-05-26 14:47:57.62
```



```
SELECT first_name,last_name FROM actor;
```

```
   first_name                 last_name
   character varying (45)     character varying (45)
1  Penelope                   Guiness
2  Nick                       Wahlberg
3  Ed
```

```
Successfully run. Total query runtime: 44 msec. 200 rows affected.
```



```
SELECT first_name,last_name FROM actor;
```

```
   first_name                 last_name
   character varying (45)     character varying (45)
1  Penelope                   Guiness
2  Nick                       Wahlberg
3  Ed
```

```
Successfully run. Total query runtime: 44 msec. 200 rows affected.
```

Q & A

SQL

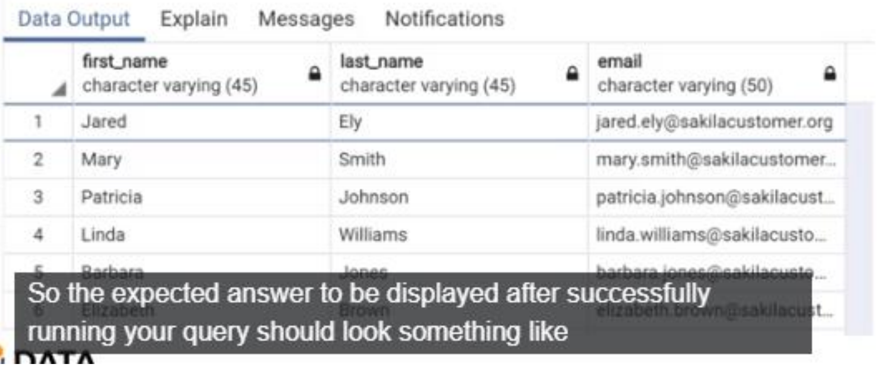
- Situation
  - We want to send out a promotional email to our existing customers!

SQL

- Challenge
  - Use a **SELECT** statement to grab the first and last names of every customer and their email address.

SQL

- Expected Answer: (may not be displayed i the exact same order)



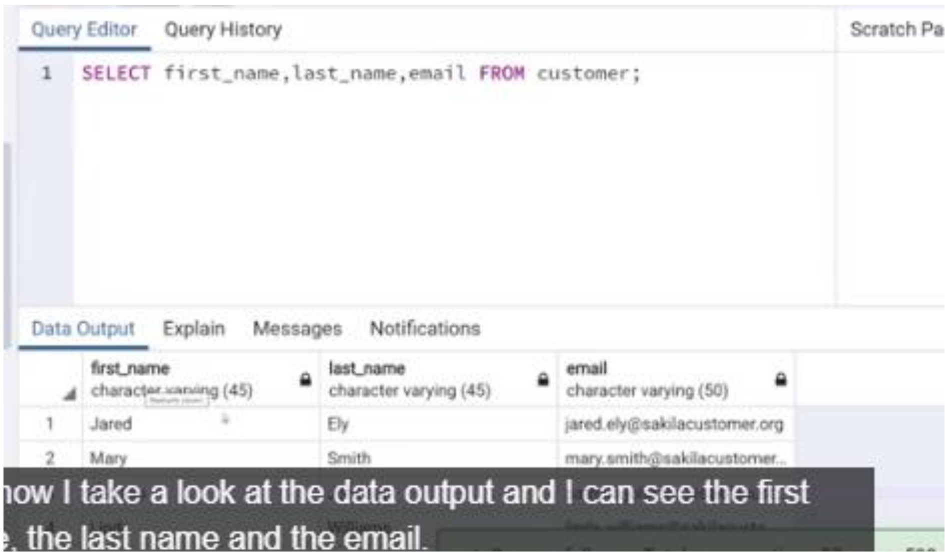
	first_name character varying (45)	last_name character varying (45)	email character varying (50)
1	Jared	Ely	jared.ely@sakilacustomer.org
2	Mary	Smith	mary.smith@sakilacustomer...
3	Patricia	Johnson	patricia.johnson@sakilacust...
4	Linda	Williams	linda.williams@sakilacusto...
5	Barbara	Jones	barbara.jones@sakilacusto...
6	Elizabeth	Brown	elizabeth.brown@sakilacust...

So the expected answer to be displayed after successfully running your query should look something like

PIERIAN DATA

SQL

- Hints
  - Use the **customer** table
  - You can use the table drop-down to view what columns are available
  - You could also use **SELECT * FROM customer** to see all the columns.



```
SELECT first_name,last_name,email FROM customer;
```

	first_name character varying (45)	last_name character varying (45)	email character varying (50)
1	Jared	Ely	jared.ely@sakilacustomer.org
2	Mary	Smith	mary.smith@sakilacustomer...

ow I take a look at the data output and I can see the first
, the last name and the email.

DISTINCT:

SQL

- Sometimes a table contains a column that has duplicate values, and you may find yourself in a situation where you only want to list the unique/distinct values.
- The **DISTINCT** keyword can be used to return only the distinct values in a column.

SQL

- The **DISTINCT** keyword operates *on* a column. The syntax looks like this:

SELECT DISTINCT column **FROM** table

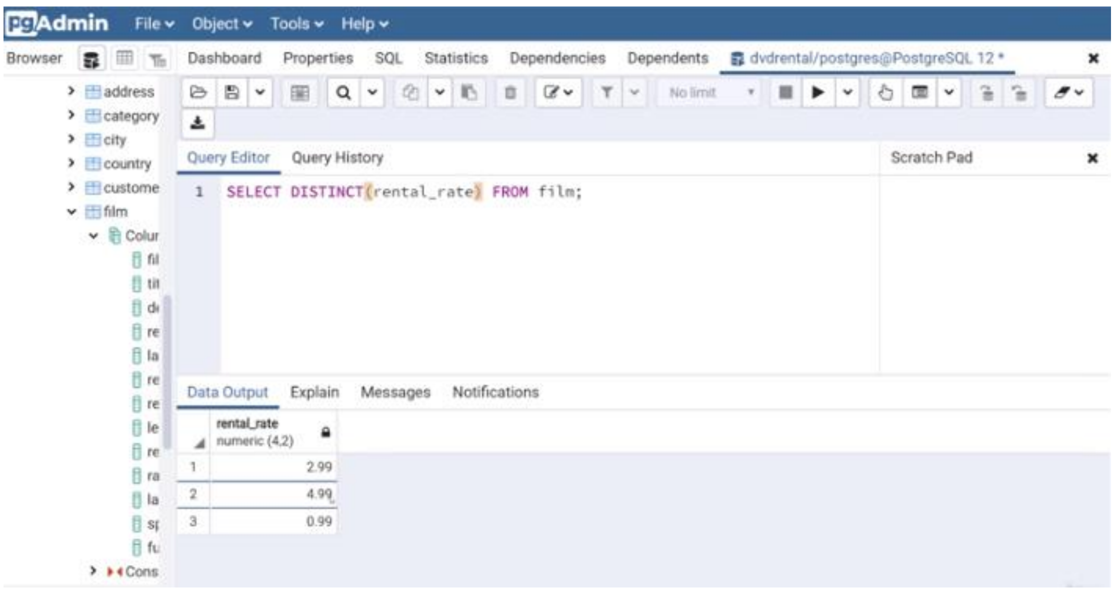
SQL

- SELECT DISTINCT name FROM color_table

Name	Choice
Zach	Green
David	Green
Claire	Yellow
David	Red

SQL

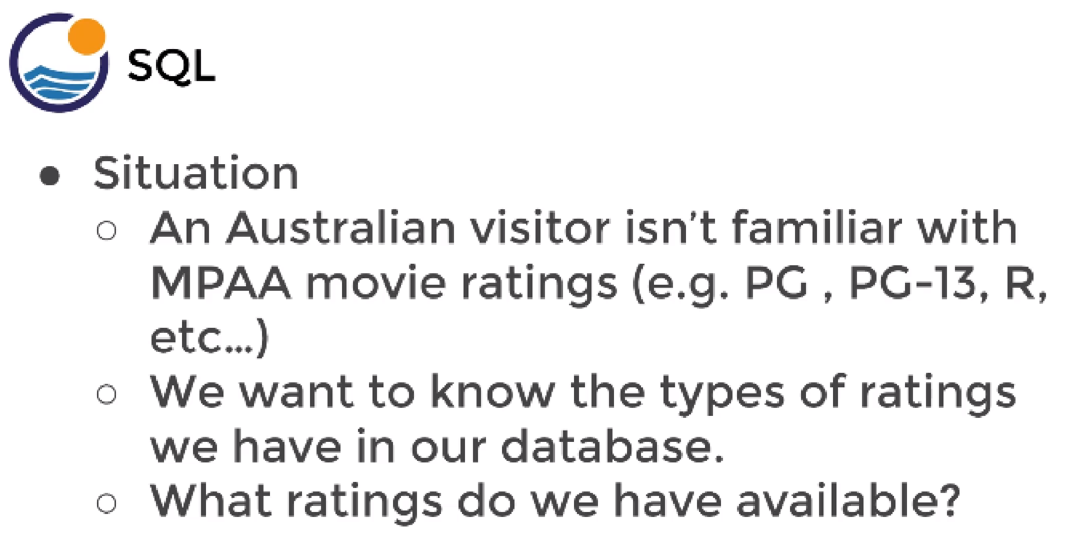
- Given the previous example, we don't really know if the person with the name "David" was a duplicate entry, or two different people with the same first name.
- Calling DISTINCT here answered the question
  - *What are the unique first names are there in the table?*



```
SELECT DISTINCT(rental_rate) FROM film;
```

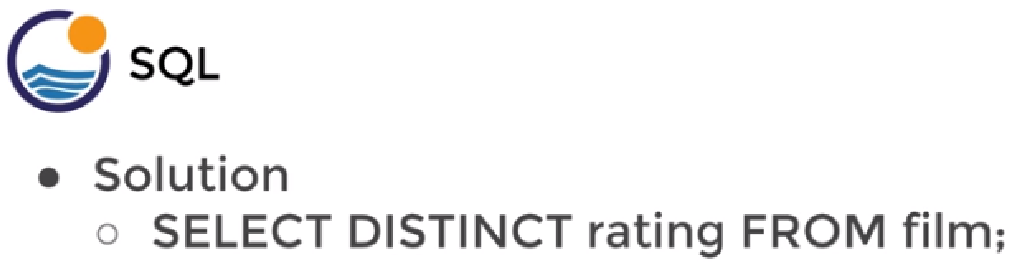
```
   rental_rate
   numeric (4,2)
1          2.99
2          4.99
3          0.99
```

Q & A:



SQL

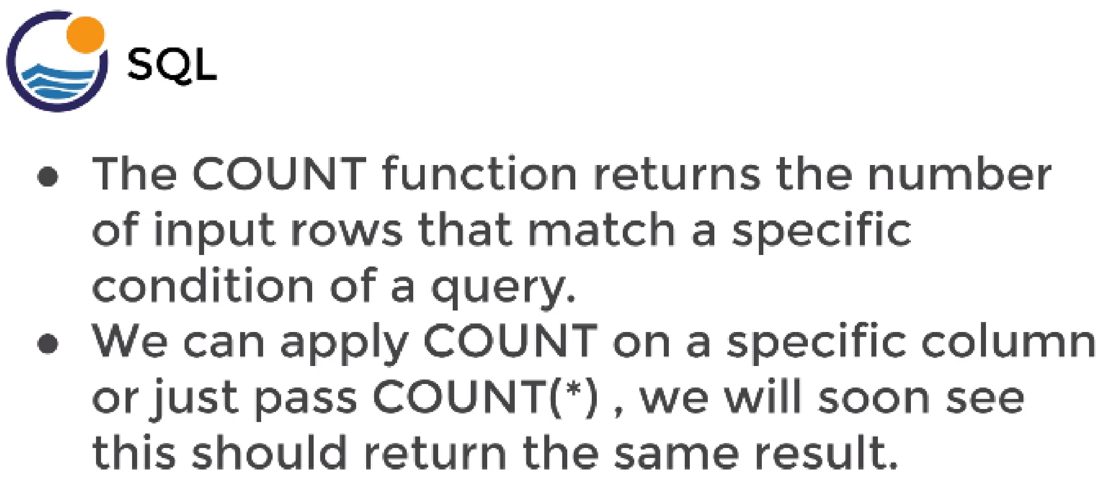
- Situation
  - An Australian visitor isn't familiar with MPAA movie ratings (e.g. PG , PG-13, R, etc...)
  - We want to know the types of ratings we have in our database.
  - What ratings do we have available?



SQL

- Solution
  - SELECT DISTINCT rating FROM film;

COUNT:



SQL

- The COUNT function returns the number of input rows that match a specific condition of a query.
- We can apply COUNT on a specific column or just pass COUNT(*) , we will soon see this should return the same result.

SQL

- SELECT COUNT(name) FROM table;

Name	Choice
Zach	Green
David	Green
Claire	Yellow
David	Red

SQL

- SELECT COUNT(name) FROM table;

Count
4

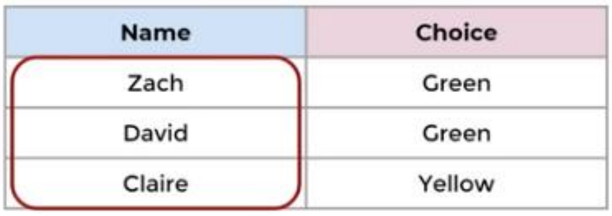
SQL

- SELECT COUNT(name) FROM table;
- SELECT COUNT(choice) FROM table;
- SELECT COUNT(*) FROM table;
- All return the same thing, since the original table had 4 rows.

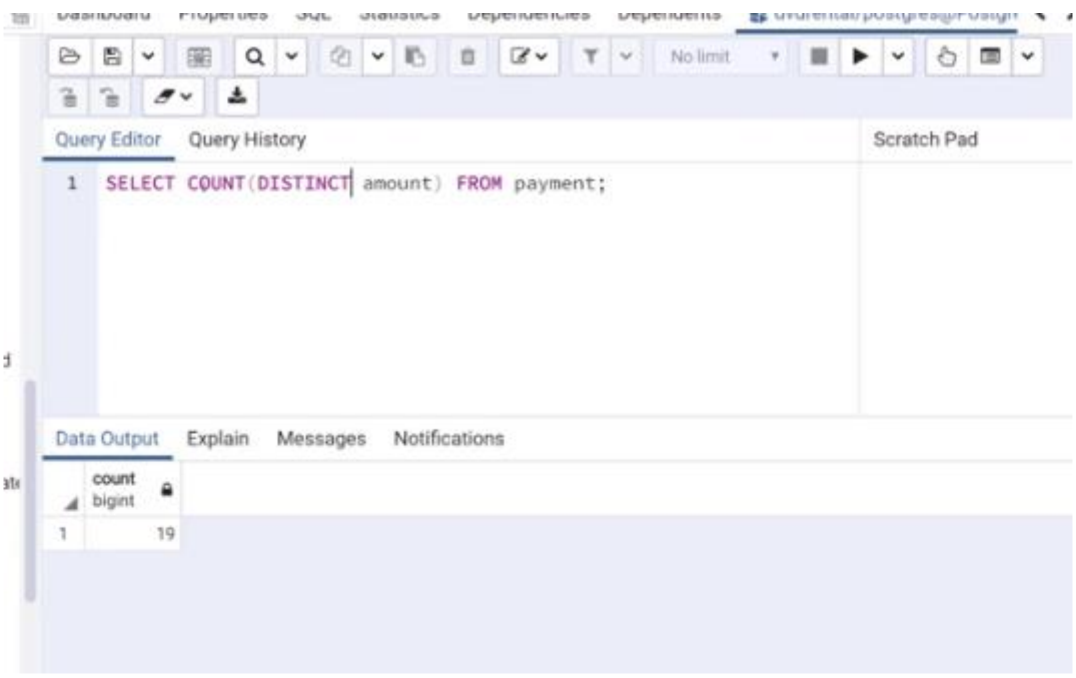
Count
4

SQL

- SELECT COUNT(DISTINCT name) FROM table;



Name	Choice
Zach	Green
David	Green
Claire	Yellow



```
SELECT COUNT(DISTINCT amount) FROM payment;
```

```
   count
   bigint
1      19
```

WHERE :

SQL

- **SELECT** and **WHERE** are the most fundamental SQL statements and you will find yourself using them often!
- The WHERE statement allows us to specify conditions on columns for the rows to be returned.

SQL

- Basic syntax example:
    - SELECT column1, column2

      FROM table

      WHERE conditions;

SQL

- The **WHERE** clause appears immediately after the FROM clause of the SELECT statement.
- The conditions are used to filter the rows returned from the SELECT statement.
- PostgreSQL provides a variety of standard operators to construct the conditions

SQL

- Comparison Operators

Operator	Description
=	Equal
>	Greater than
<	Less Than
>=	Greater than or equal to
<=	Less than or equal to
<> or !=	Not equal to

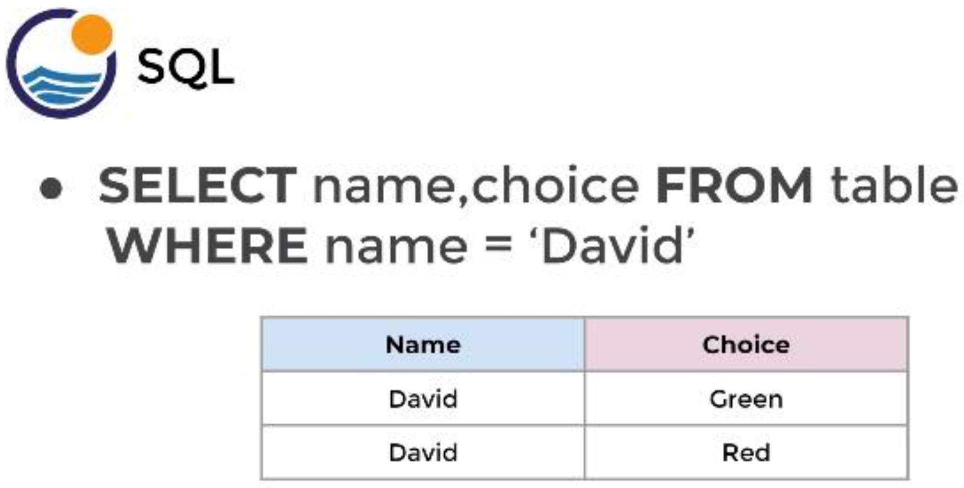
SQL

- Logical Operators
  - Allow us to combine multiple comparison operators
    - **AND**
    - **OR**
    - **NOT**

SQL

- **SELECT** name,choice **FROM** table
- Now let's get only the people named David

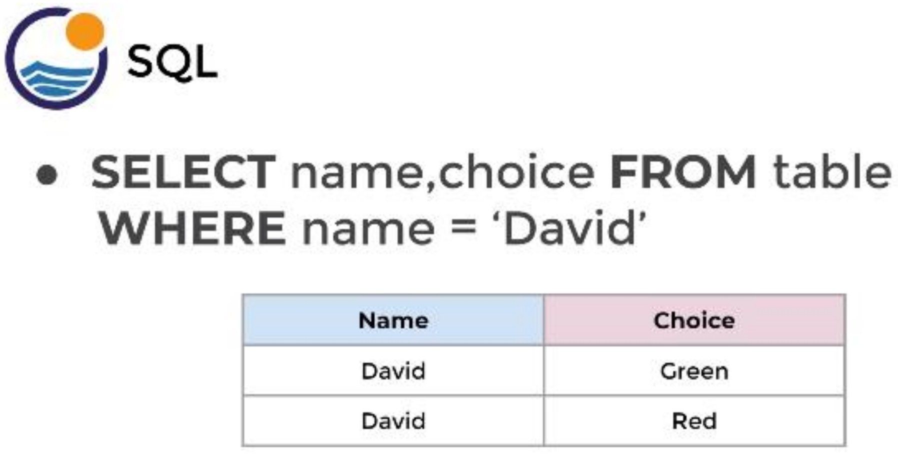
Name	Choice
Zach	Green
David	Green
Claire	Yellow
David	Red



SQL

- **SELECT** name,choice **FROM** table **WHERE** name = ‘David’

Name	Choice
David	Green
David	Red

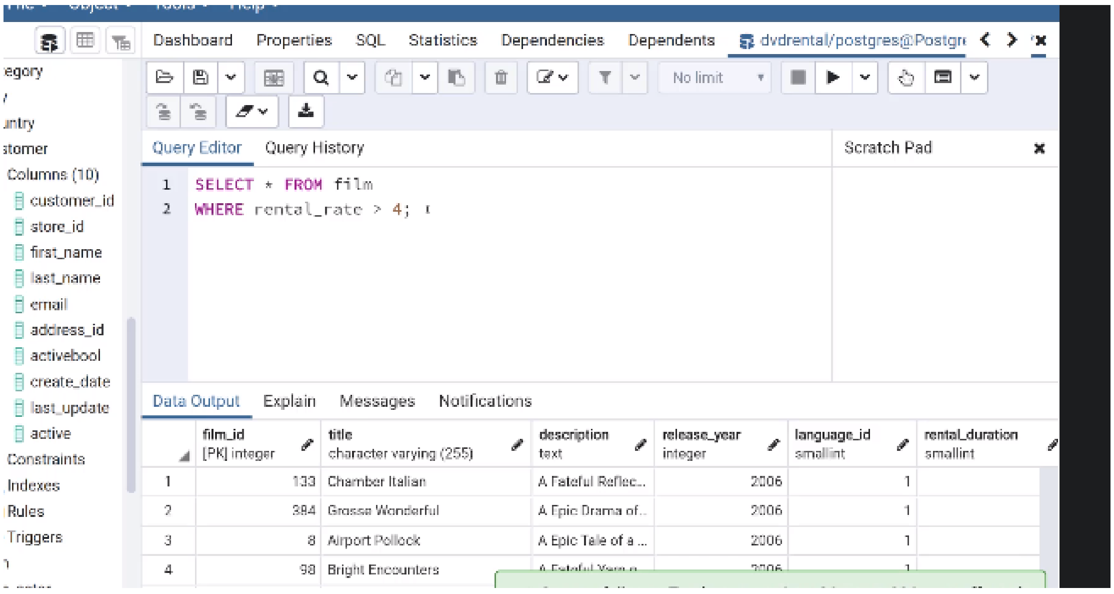


SQL

- **SELECT** name,choice **FROM** table **WHERE** name = ‘David’

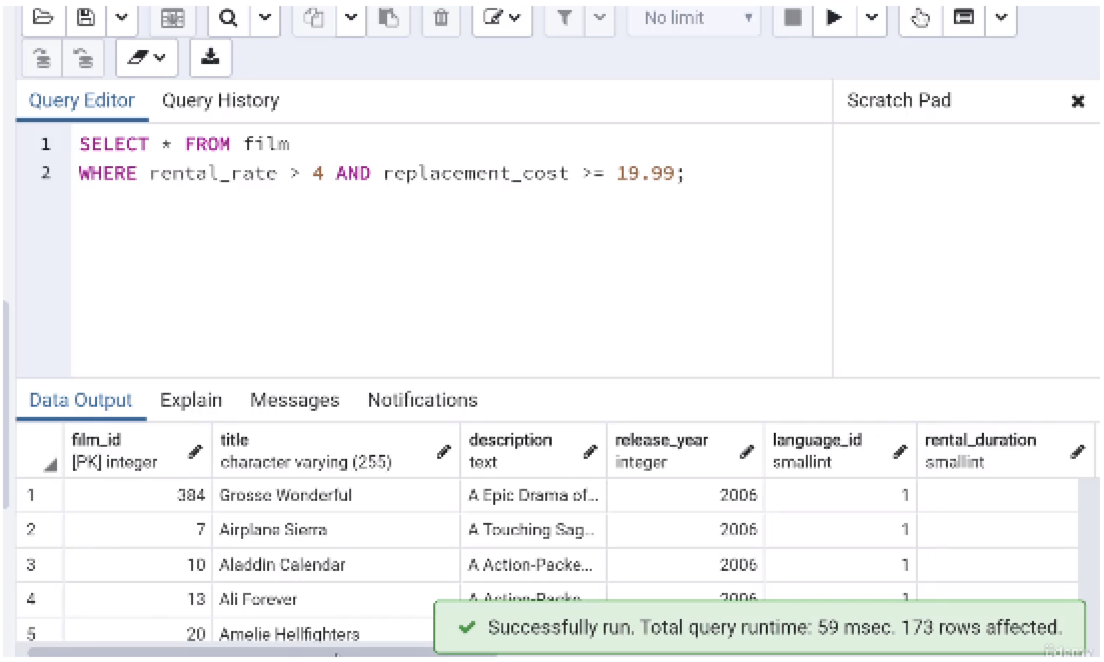
Name	Choice
David	Green
David	Red

SELECT WHERE:



```
SELECT * FROM film
WHERE rental_rate > 4;
```

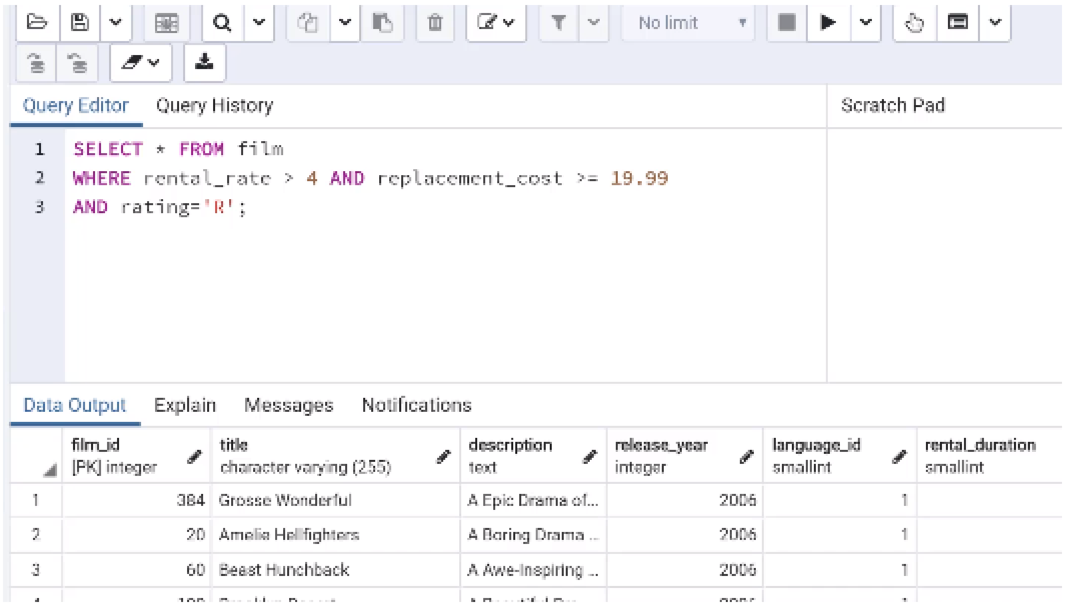
	film_id [PK] integer	title character varying (255)	description text	release_year integer	language_id smallint	rental_duration smallint
1	133	Chamber Italian	A Fateful Reflec...	2006	1	
2	384	Grosse Wonderful	A Epic Drama of...	2006	1	
3	8	Airport Pollock	A Epic Tale of a ...	2006	1	
4	98	Bright Encounters	A Fateful Yarn a...	2006	1	



```sql
SELECT * FROM film
WHERE rental_rate > 4 AND replacement_cost >= 19.99;
```

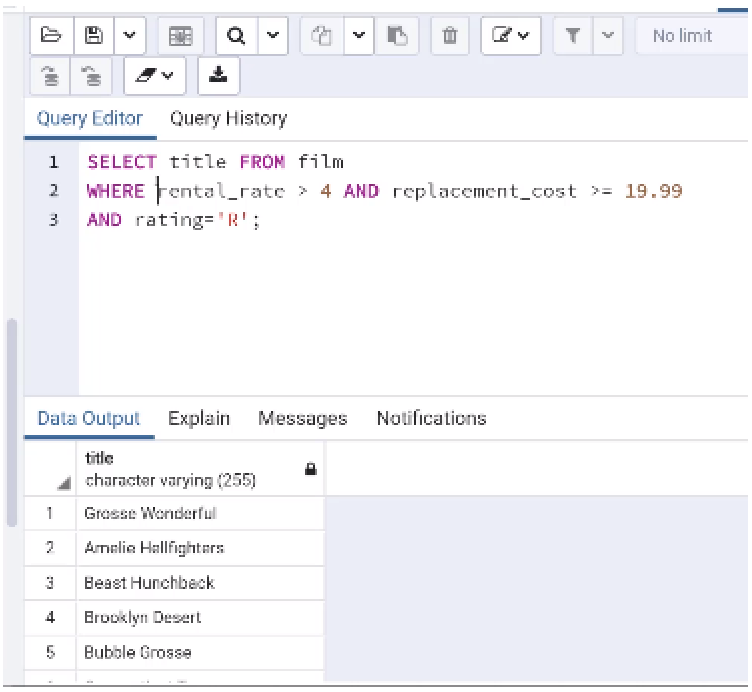
	film_id [PK] integer	title character varying (255)	description text	release_year integer	language_id smallint	rental_duration smallint
1	384	Grosse Wonderful	A Epic Drama of...	2006	1	
2	7	Airplane Sierra	A Touching Sag...	2006	1	
3	10	Aladdin Calendar	A Action-Packe...	2006	1	
4	13	Ali Forever	A Action-Packe...	2006	1	
5	20	Amelie Hellfighters				

✔ Successfully run. Total query runtime: 59 msec. 173 rows affected.



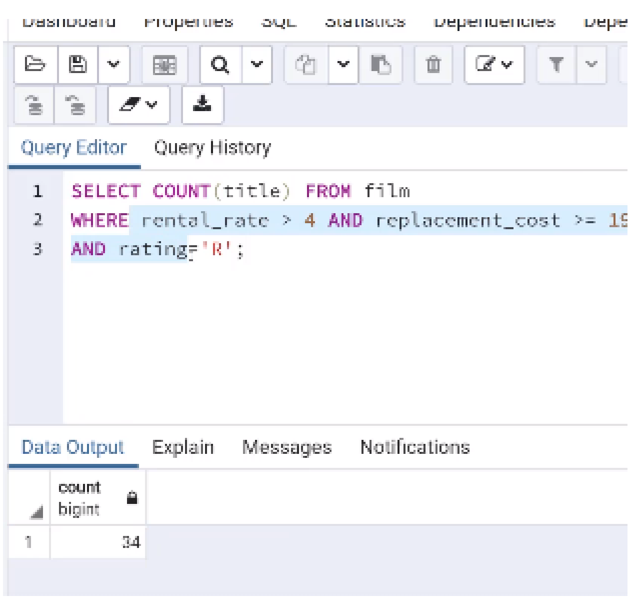
```sql
SELECT * FROM film
WHERE rental_rate > 4 AND replacement_cost >= 19.99
AND rating='R';
```

	film_id [PK] integer	title character varying (255)	description text	release_year integer	language_id smallint	rental_duration smallint
1	384	Grosse Wonderful	A Epic Drama of...	2006	1	
2	20	Amelie Hellfighters	A Boring Drama ...	2006	1	
3	60	Beast Hunchback	A Awe-Inspiring ...	2006	1	



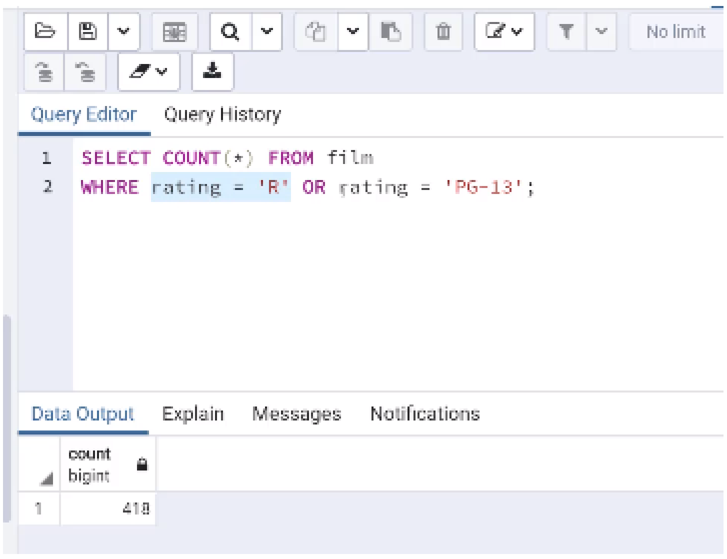
```
SELECT title FROM film
WHERE rental_rate > 4 AND replacement_cost >= 19.99
AND rating='R';
```

	title character varying (255)
1	Grosse Wonderful
2	Amelie Hellfighters
3	Beast Hunchback
4	Brooklyn Desert
5	Bubble Grosse



```
SELECT COUNT(title) FROM film
WHERE rental_rate > 4 AND replacement_cost >= 19
AND rating='R';
```

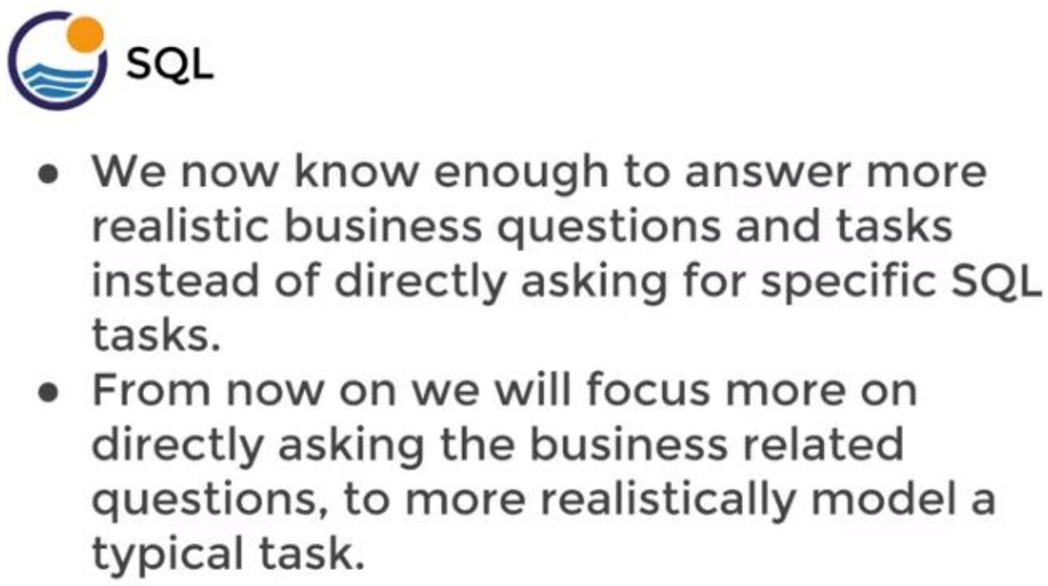
	count bigint
1	34



```sql
SELECT COUNT(*) FROM film
WHERE rating = 'R' OR rating = 'PG-13';
```

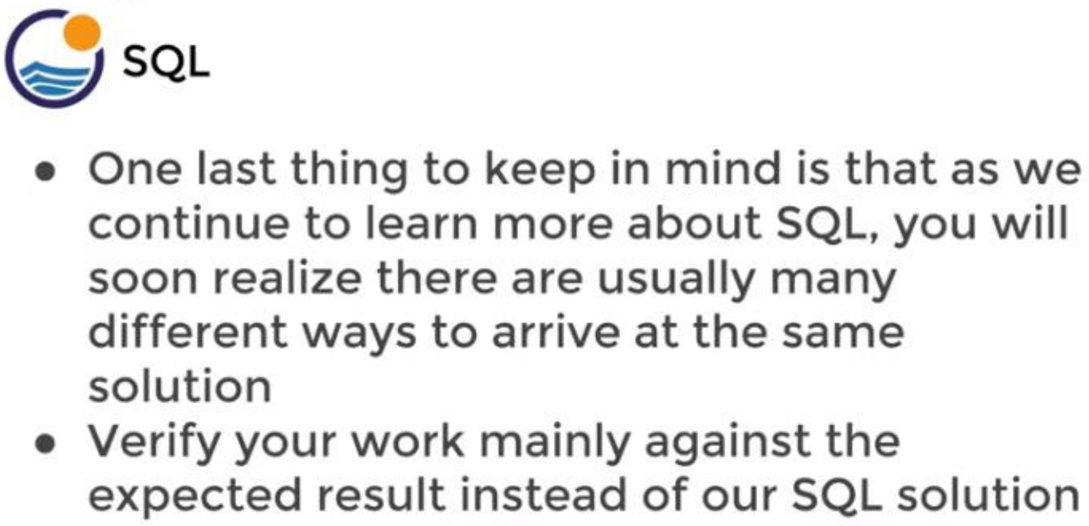
```
  count
  bigint
1    418
```

Q & A



SQL

- We now know enough to answer more realistic business questions and tasks instead of directly asking for specific SQL tasks.
- From now on we will focus more on directly asking the business related questions, to more realistically model a typical task.



SQL

- One last thing to keep in mind is that as we continue to learn more about SQL, you will soon realize there are usually many different ways to arrive at the same solution
- Verify your work mainly against the expected result instead of our SQL solution

SQL

- Challenge No. 1
  - A customer forgot their wallet at our store! We need to track down their email to inform them.
  - What is the email for the customer with the name Nancy Thomas?

SQL

- Solution for Challenge No. 1
  - SELECT email FROM customer

    WHERE first_name = 'Nancy'

    AND last_name = 'Thomas';

SQL

- Challenge No. 2
  - A customer wants to know what the movie "Outlaw Hanky" is about.
  - Could you give them the description for the movie "Outlaw Hanky"?

SQL

- Solution for Challenge No. 2
  - SELECT description FROM film

    WHERE title = 'Outlaw Hanky';

SQL

- Challenge No. 3
  - A customer is late on their movie return, and we've mailed them a letter to their address at **'259 Ipoh Drive'**. We should also call them on the phone to let them know.
  - Can you get the phone number for the customer who lives at **'259 Ipoh Drive'?**

SQL

- Solution for Challenge No. 3
  - SELECT phone FROM address

    WHERE address= '259 Ipoh Drive';