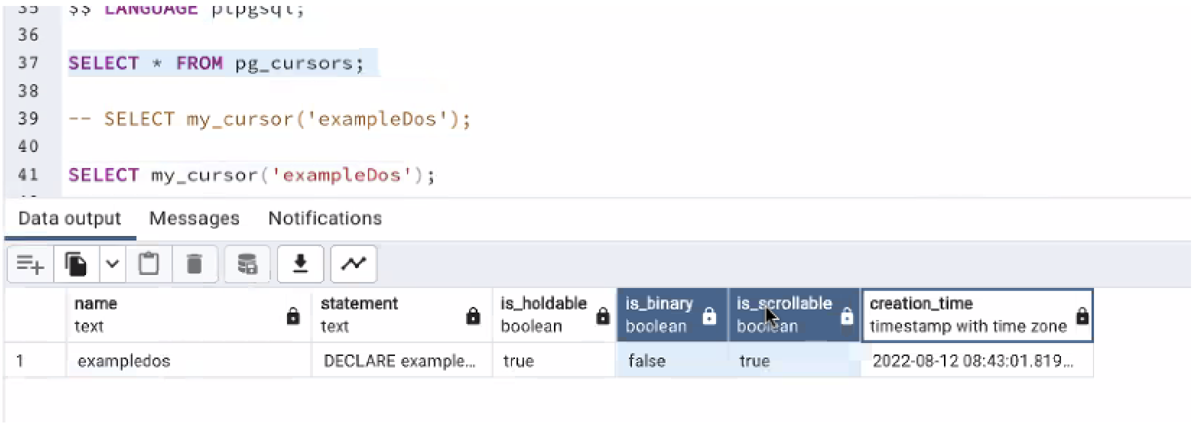
Examen de postgres y bash dentro de un futuro

Cursor

Good practice to create it within transactions
You can select the cursor to see more info

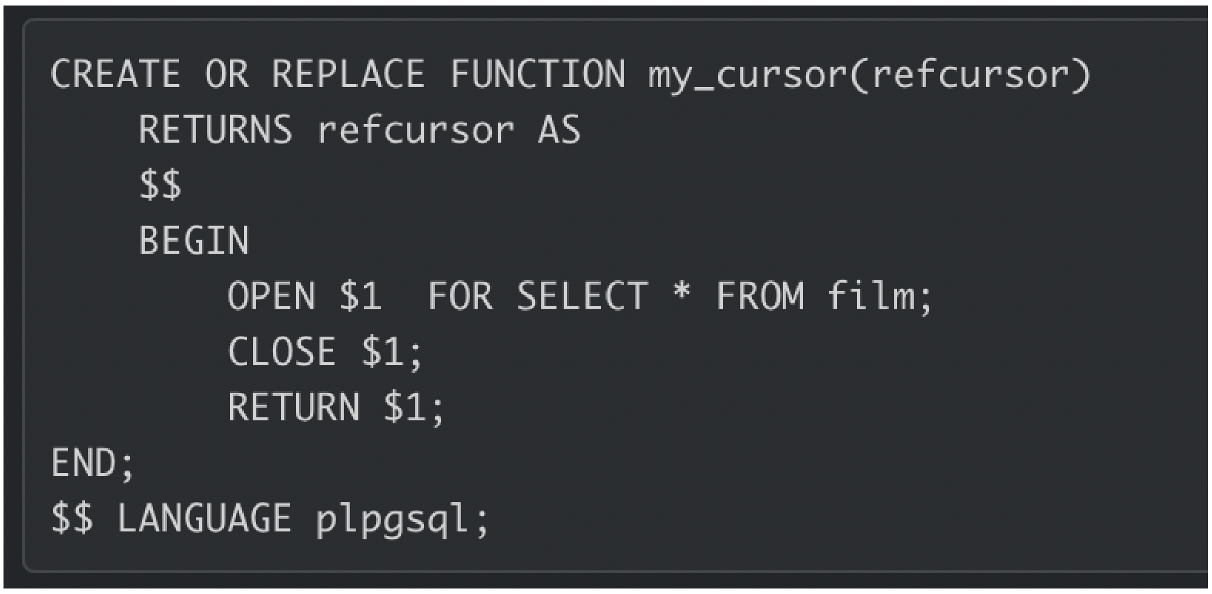


```
$$ LANGUAGE plpgsql;

SELECT * FROM pg_cursors;

-- SELECT my_cursor('exampleDos');

SELECT my_cursor('exampleDos');
```

	name text	statement text	is_holdable boolean	is_binary boolean	is_scrollable boolean	creation_time timestamp with time zone
1	exampledos	DECLARE example...	true	false	true	2022-08-12 08:43:01.819...

He used refcursor

Declarar cursores dentro de funciones o variables ?

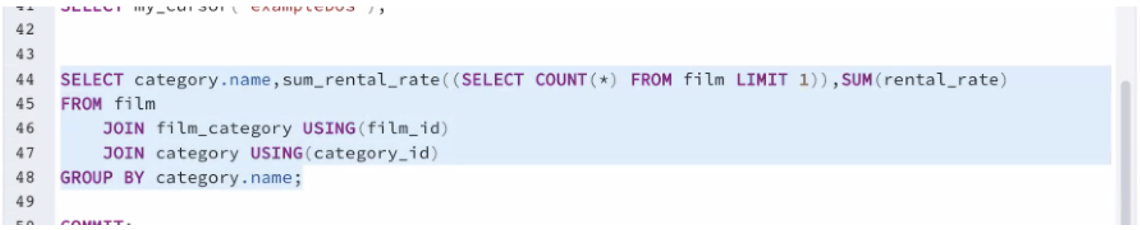


```
CREATE OR REPLACE FUNCTION my_cursor(refcursor)
    RETURNS refcursor AS
    $$
    BEGIN
        OPEN $1  FOR SELECT * FROM film;
        CLOSE $1;
        RETURN $1;
END;
$$ LANGUAGE plpgsql;
```

La manera correcta de usar un cursor es unirlo a un query.
Y usando una función ahí lo obligamos a que esté unido.

Nested aggregate

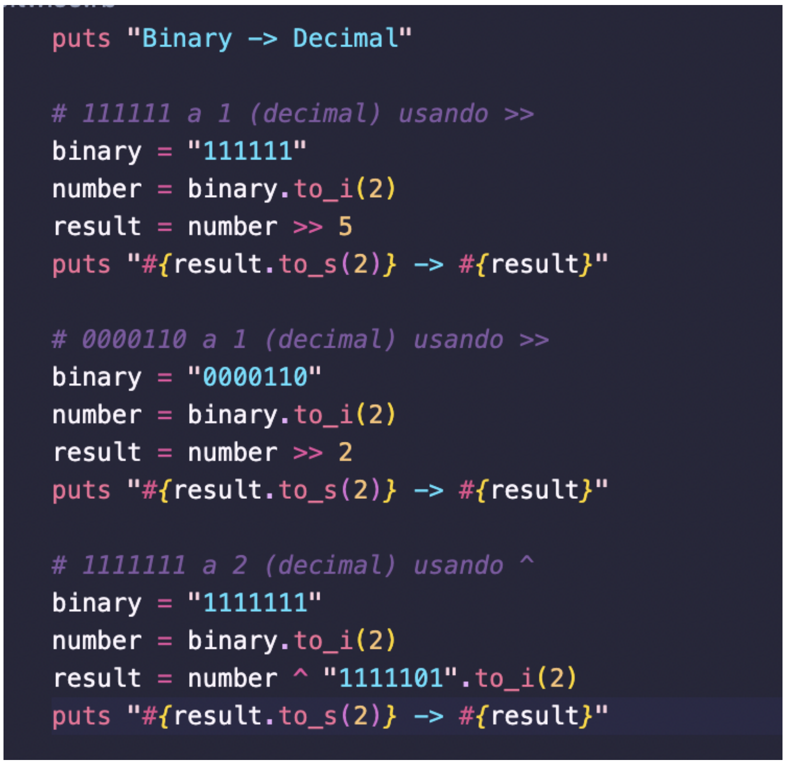
The only way to do them is with a nested select.



```sql
SELECT my_cursor( 'exampledos' );


SELECT category.name,sum_rental_rate((SELECT COUNT(*) FROM film LIMIT 1)),SUM(rental_rate)
FROM film
    JOIN film_category USING(film_id)
    JOIN category USING(category_id)
GROUP BY category.name;

COMMIT;
```

Bitwise operators

Usados para operaciones matemáticas binarias.



```ruby
puts "Binary -> Decimal"

# 111111 a 1 (decimal) usando >>
binary = "111111"
number = binary.to_i(2)
result = number >> 5
puts "#{result.to_s(2)} -> #{result}"

# 0000110 a 1 (decimal) usando >>
binary = "0000110"
number = binary.to_i(2)
result = number >> 2
puts "#{result.to_s(2)} -> #{result}"

# 1111111 a 2 (decimal) usando ^
binary = "1111111"
number = binary.to_i(2)
result = number ^ "1111101".to_i(2)
puts "#{result.to_s(2)} -> #{result}"
```