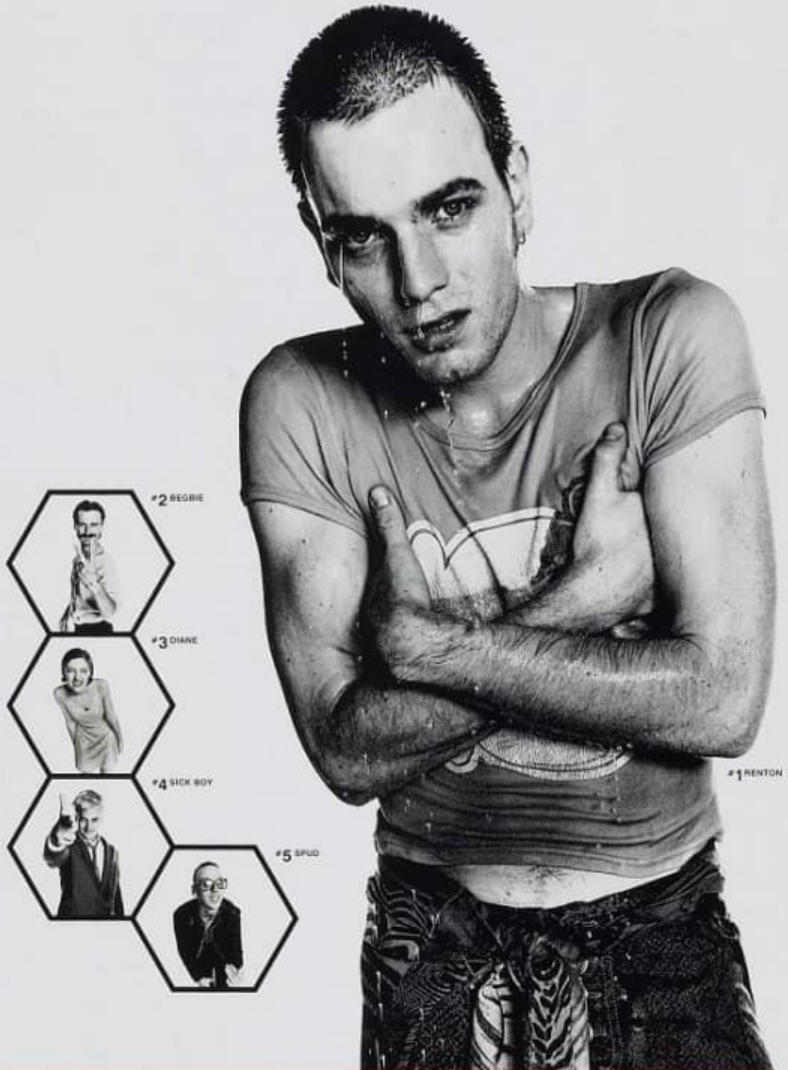

SELECT
'amazing_features'
FROM
"postgresql"

# Postgres **95**

# PostgreSQL **v1**



# PostgreSQL **v6...**

# Kevin DAVIN

@davinkevin@davinkevin.fr

Google Developer Expert
on **Google Cloud** & **Kotlin**

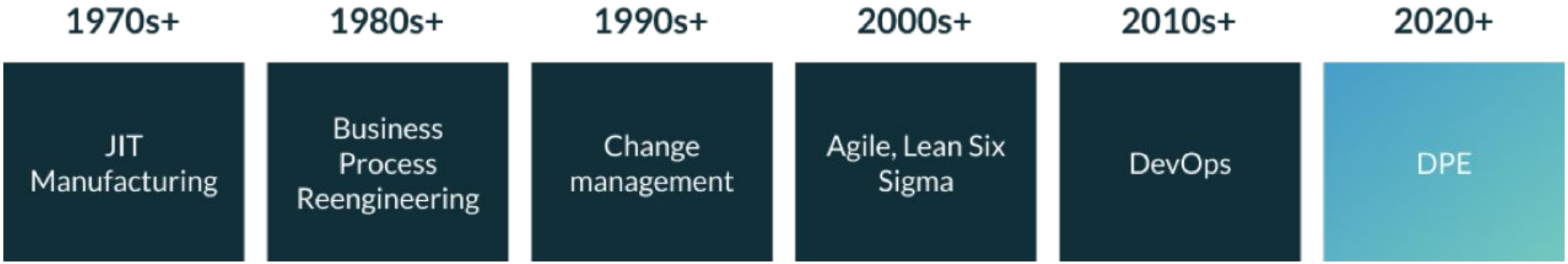
Gitlab Heroes

Open Source Contributor


Gradle

Our mission at **Gradle** is to **accelerate developer productivity** and **make developers happier**

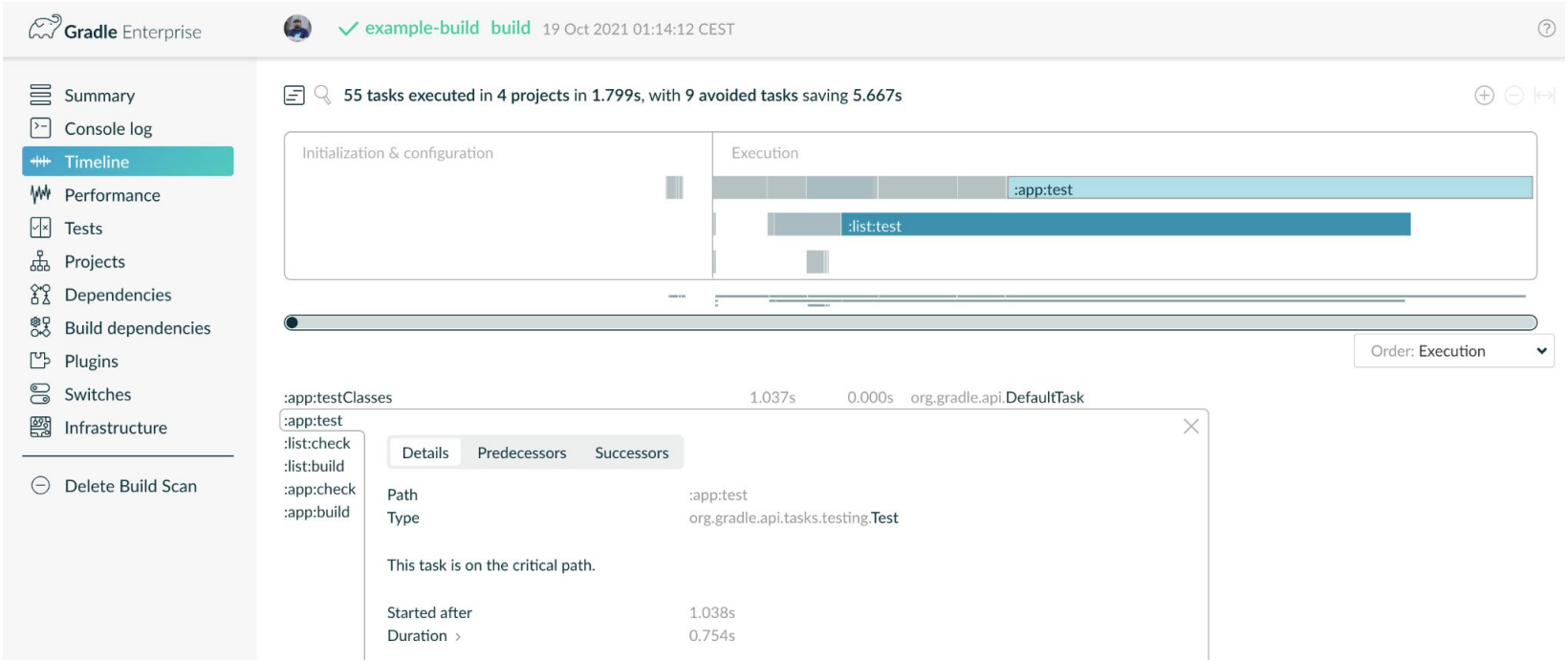
What comes after DevOps?



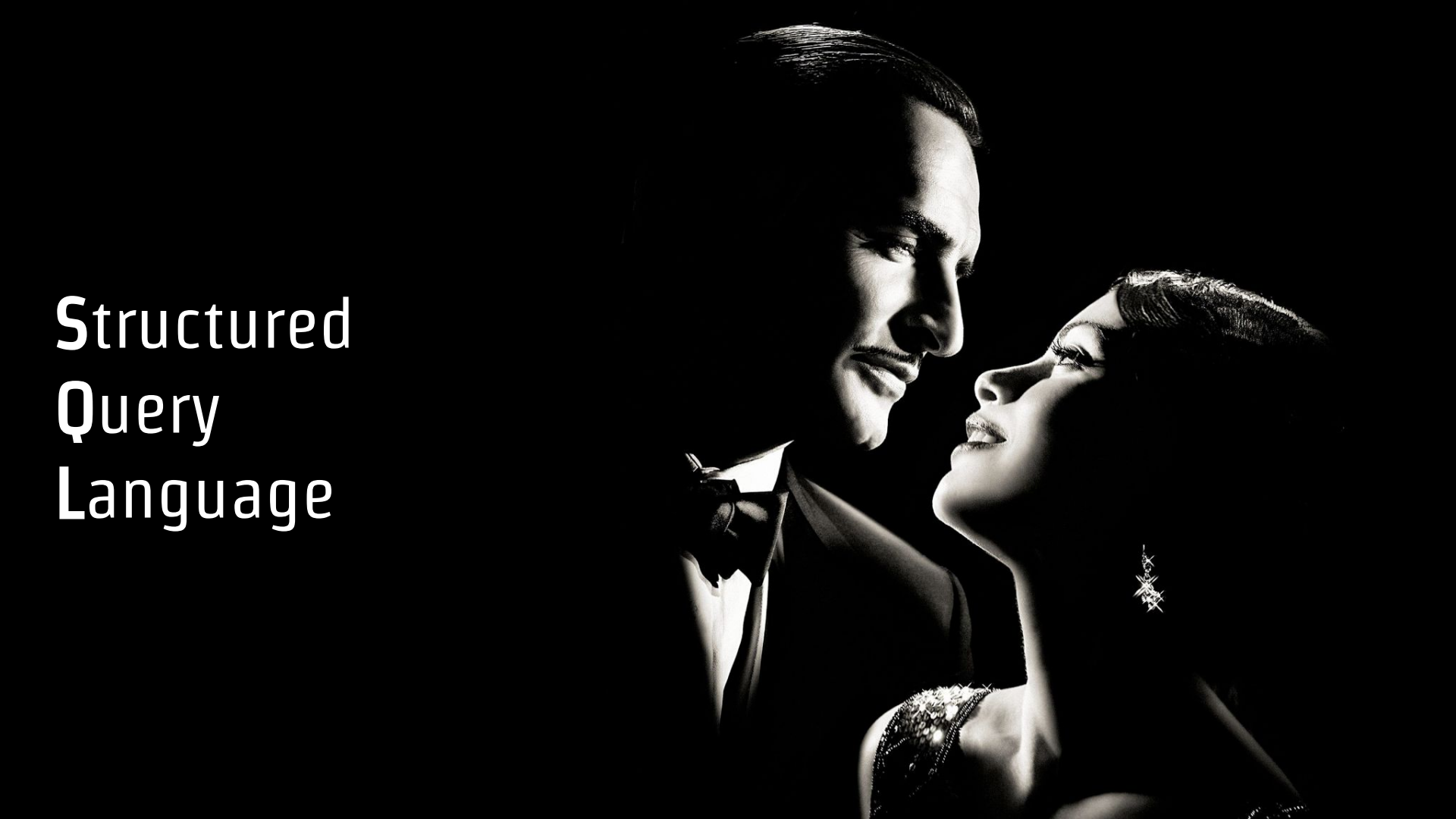

**D**eveloper **P**roductivity **E**ngineering

# “If you can’t measure it, you can’t improve it!”

*by Peter Drucker*



**S**tructured
**Q**uery
**L**anguage



**Standard**

ISO/IEC **9075-1**

SQL:1986

# SQL:2016

SQL:2023

Never
Forget
The order...

From
Join

Where

Group by

Aggregate
Functions

Having

Window
Functions

Select

Distinct

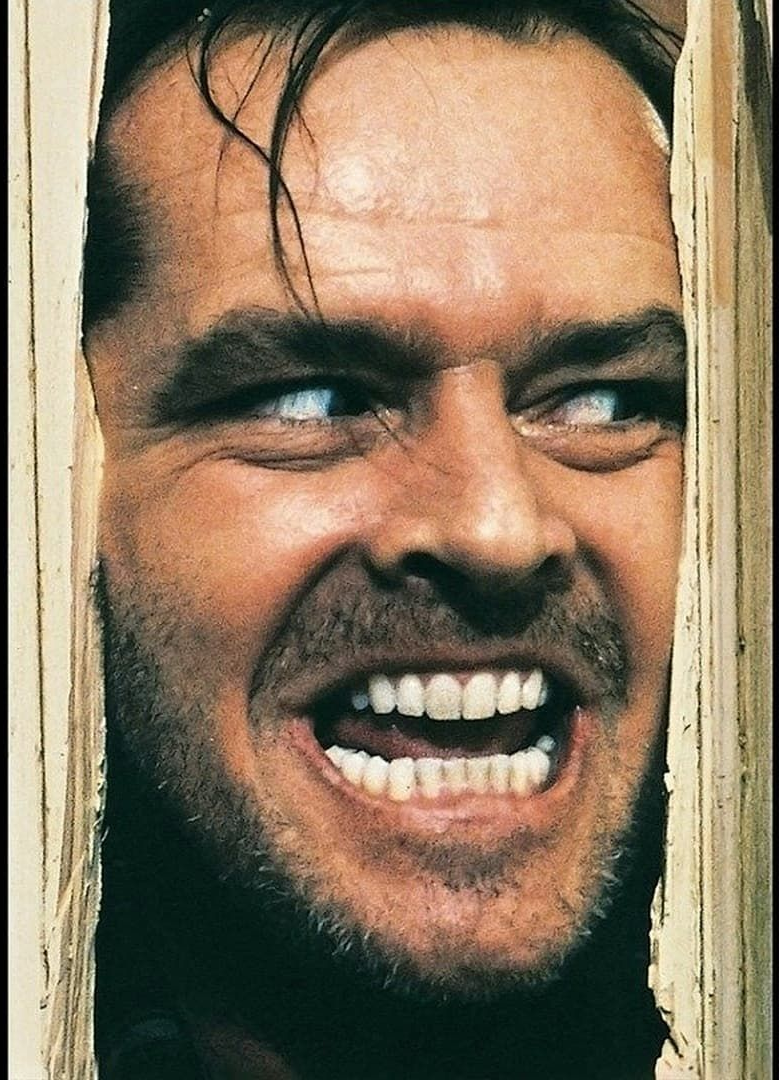
Union
Intersect
Except

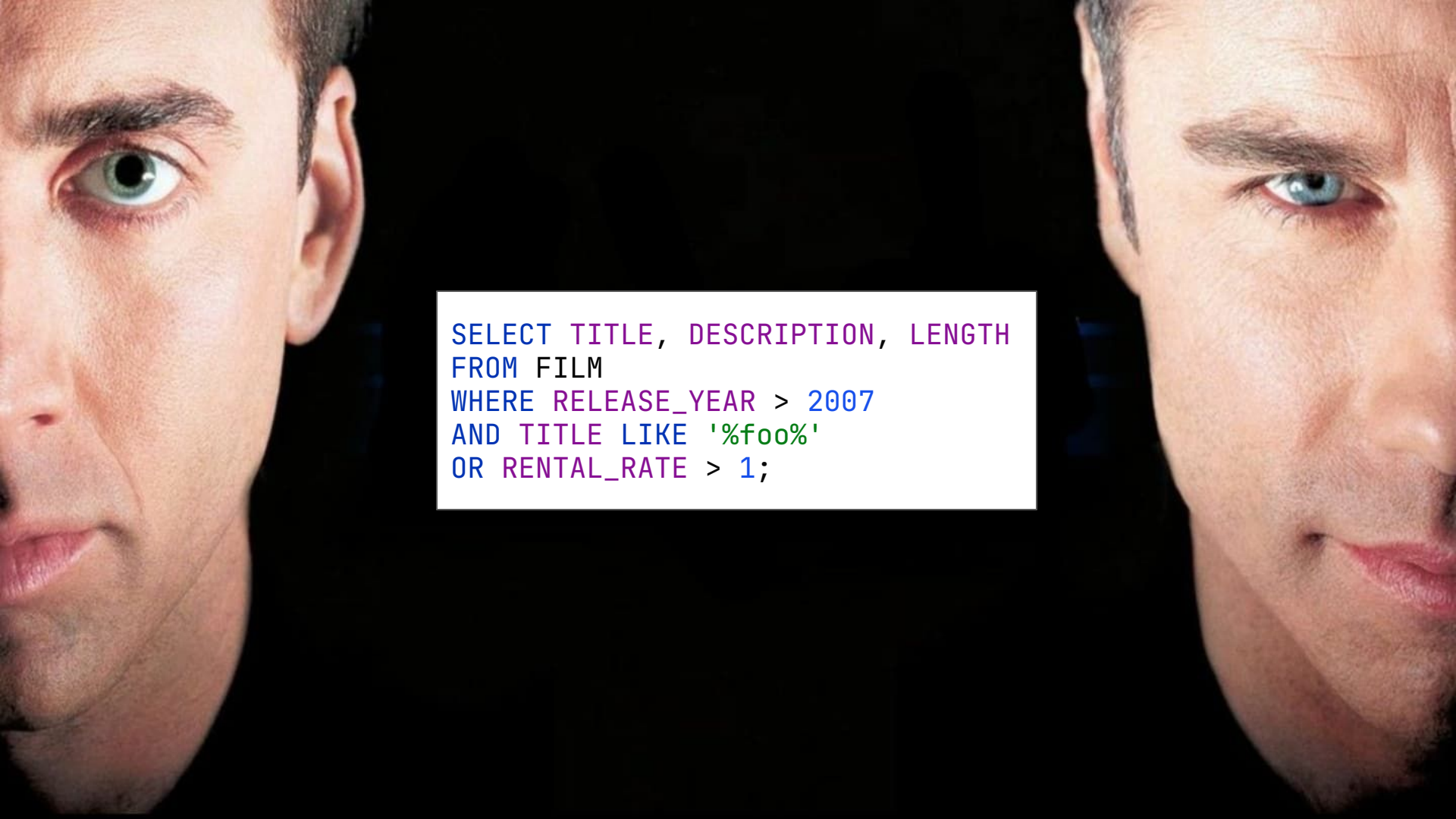
Order by

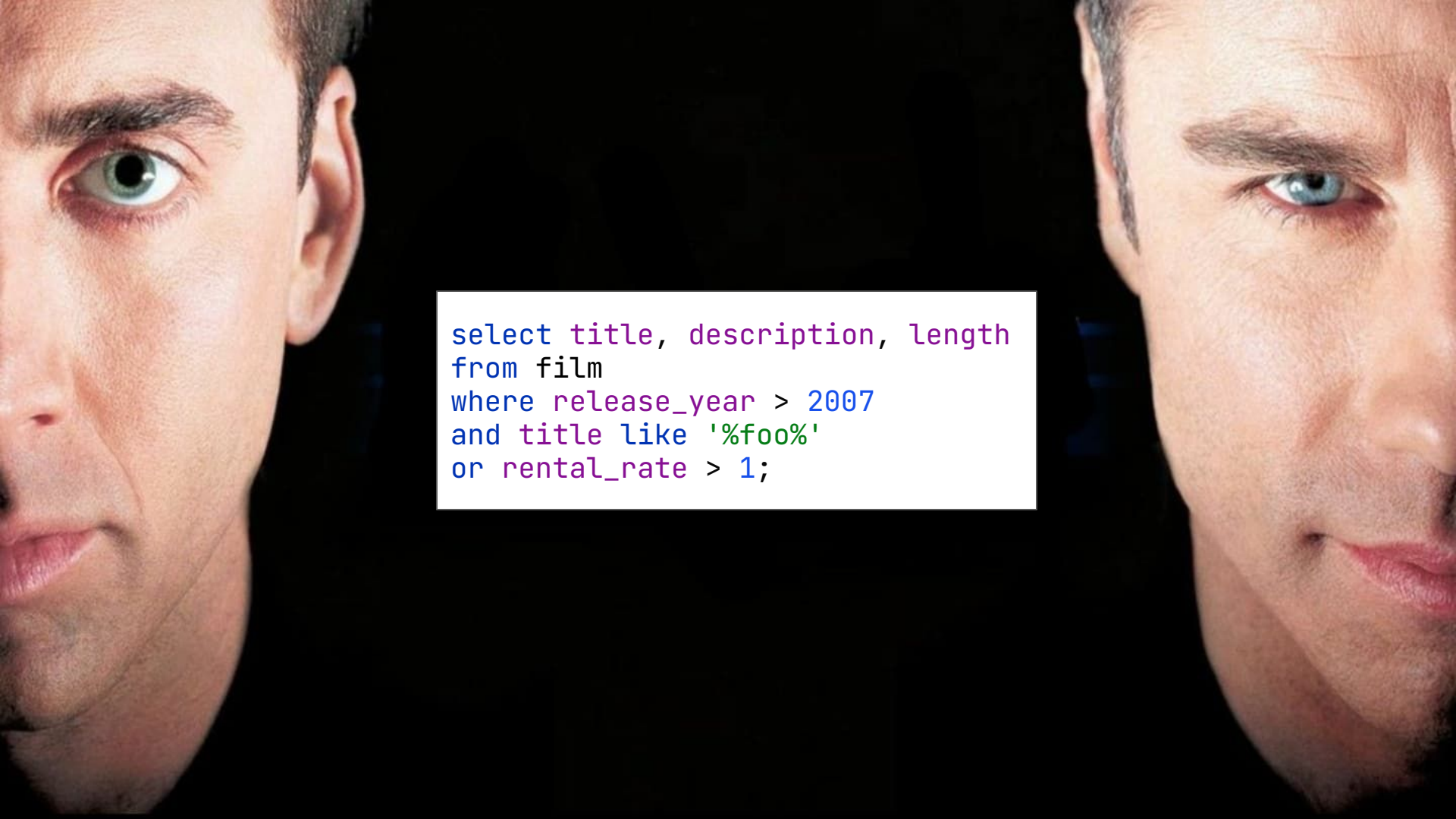
Offset

Limit
Fetch
Top

Don’t need to shout…

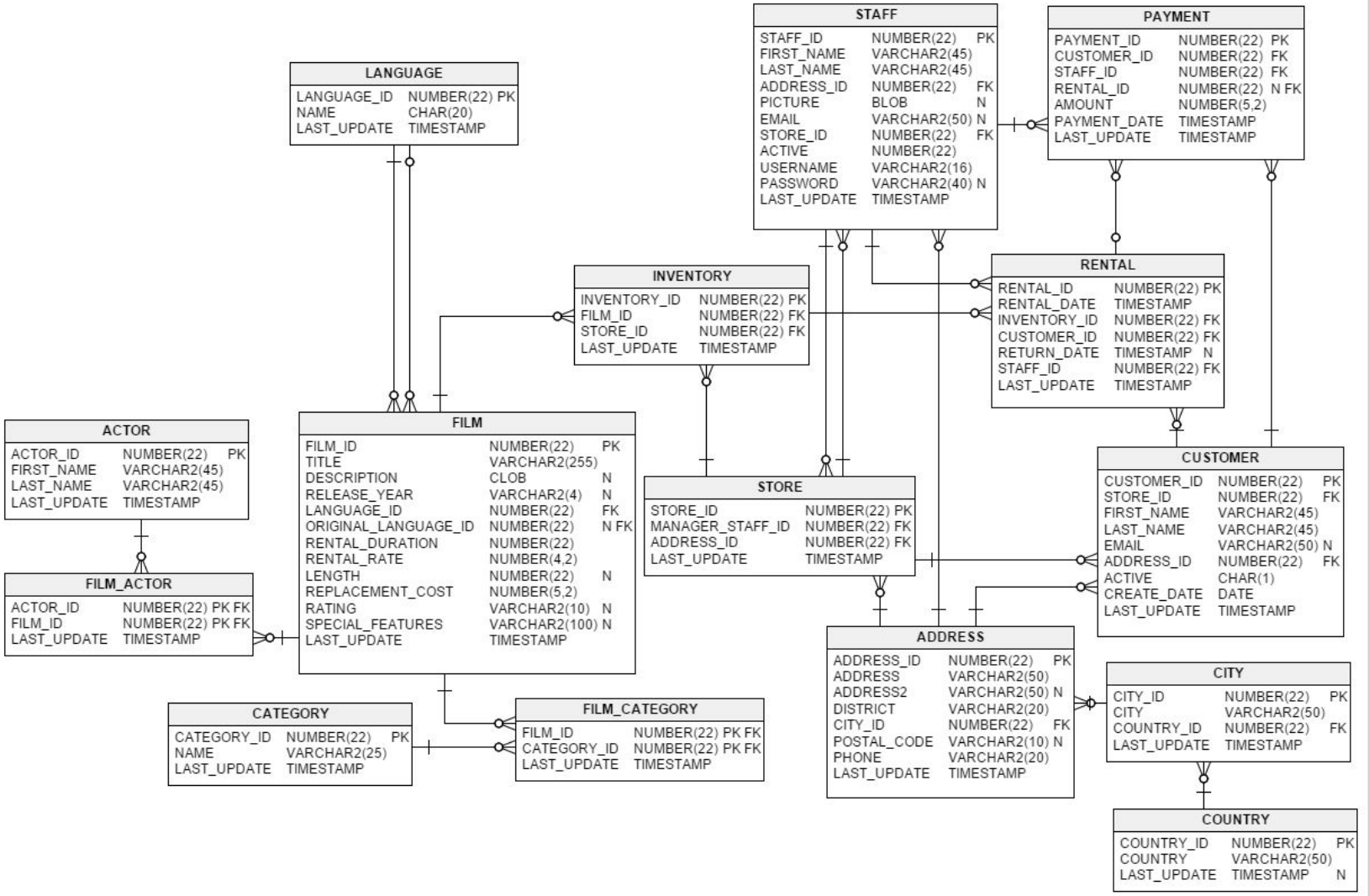


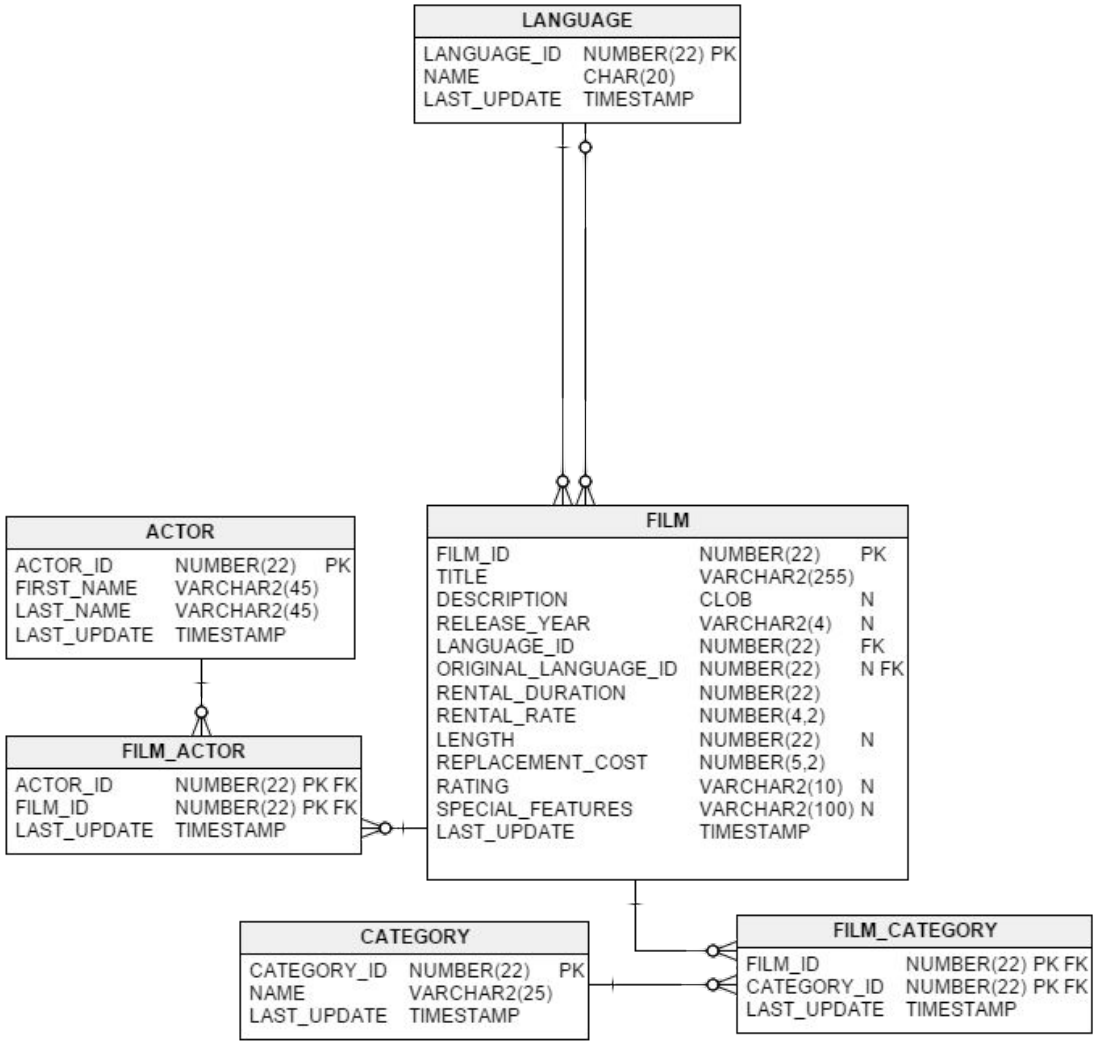

SELECT TITLE, DESCRIPTION, LENGTH
FROM FILM
WHERE RELEASE_YEAR > 2007
AND TITLE LIKE '%foo%'
OR RENTAL_RATE > 1;

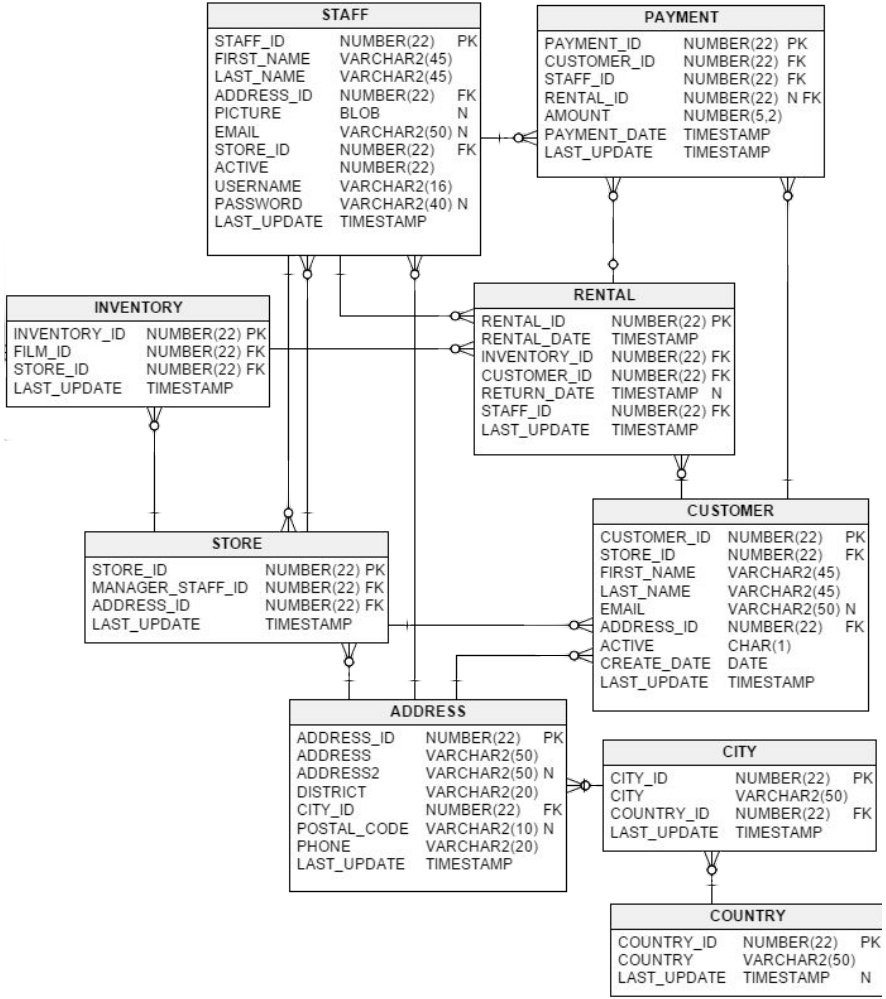


```sql
select title, description, length
from film
where release_year > 2007
and title like '%foo%'
or rental_rate > 1;
```

# Sakila Database

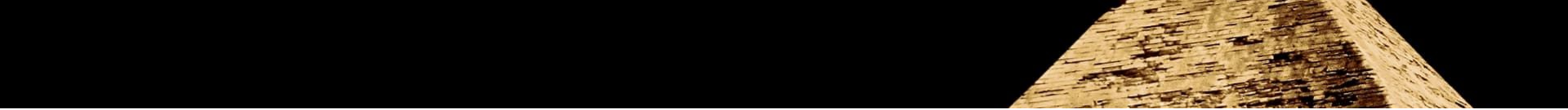

STAFF
STAFF_ID NUMBER(22) PK
FIRST_NAME VARCHAR2(45)
LAST_NAME VARCHAR2(45)
ADDRESS_ID NUMBER(22) FK
PICTURE BLOB N
EMAIL VARCHAR2(50) N
STORE_ID NUMBER(22) FK
ACTIVE NUMBER(22)
USERNAME VARCHAR2(16)
PASSWORD VARCHAR2(40) N
LAST_UPDATE TIMESTAMP
PAYMENT
PAYMENT_ID NUMBER(22) PK
CUSTOMER_ID NUMBER(22) FK
STAFF_ID NUMBER(22) FK
RENTAL_ID NUMBER(22) N FK
AMOUNT NUMBER(5,2)
PAYMENT_DATE TIMESTAMP
LAST_UPDATE TIMESTAMP
LANGUAGE
LANGUAGE_ID NUMBER(22) PK
NAME CHAR(20)
LAST_UPDATE TIMESTAMP
INVENTORY
INVENTORY_ID NUMBER(22) PK
FILM_ID NUMBER(22) FK
STORE_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
RENTAL
RENTAL_ID NUMBER(22) PK
RENTAL_DATE TIMESTAMP
INVENTORY_ID NUMBER(22) FK
CUSTOMER_ID NUMBER(22) FK
RETURN_DATE TIMESTAMP N
STAFF_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
ACTOR
ACTOR_ID NUMBER(22) PK
FIRST_NAME VARCHAR2(45)
LAST_NAME VARCHAR2(45)
LAST_UPDATE TIMESTAMP
FILM
FILM_ID NUMBER(22) PK
TITLE VARCHAR2(255)
DESCRIPTION CLOB N
RELEASE_YEAR VARCHAR2(4) N
LANGUAGE_ID NUMBER(22) FK
ORIGINAL_LANGUAGE_ID NUMBER(22) N FK
RENTAL_DURATION NUMBER(22)
RENTAL_RATE NUMBER(4,2)
LENGTH NUMBER(22) N
REPLACEMENT_COST NUMBER(5,2)
RATING VARCHAR2(10) N
SPECIAL_FEATURES VARCHAR2(100) N
LAST_UPDATE TIMESTAMP
STORE
STORE_ID NUMBER(22) PK
MANAGER_STAFF_ID NUMBER(22) FK
ADDRESS_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
CUSTOMER
CUSTOMER_ID NUMBER(22) PK
STORE_ID NUMBER(22) FK
FIRST_NAME VARCHAR2(45)
LAST_NAME VARCHAR2(45)
EMAIL VARCHAR2(50) N
ADDRESS_ID NUMBER(22) FK
ACTIVE CHAR(1)
CREATE_DATE DATE
LAST_UPDATE TIMESTAMP
FILM_ACTOR
ACTOR_ID NUMBER(22) PK FK
FILM_ID NUMBER(22) PK FK
LAST_UPDATE TIMESTAMP
ADDRESS
ADDRESS_ID NUMBER(22) PK
ADDRESS VARCHAR2(50)
ADDRESS2 VARCHAR2(50) N
DISTRICT VARCHAR2(20)
CITY_ID NUMBER(22) FK
POSTAL_CODE VARCHAR2(10) N
PHONE VARCHAR2(20)
LAST_UPDATE TIMESTAMP
CITY
CITY_ID NUMBER(22) PK
CITY VARCHAR2(50)
COUNTRY_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
CATEGORY
CATEGORY_ID NUMBER(22) PK
NAME VARCHAR2(25)
LAST_UPDATE TIMESTAMP
FILM_CATEGORY
FILM_ID NUMBER(22) PK FK
CATEGORY_ID NUMBER(22) PK FK
LAST_UPDATE TIMESTAMP
COUNTRY
COUNTRY_ID NUMBER(22) PK
COUNTRY VARCHAR2(50)
LAST_UPDATE TIMESTAMP N


LANGUAGE
LANGUAGE_ID NUMBER(22) PK
NAME CHAR(20)
LAST_UPDATE TIMESTAMP
ACTOR
ACTOR_ID NUMBER(22) PK
FIRST_NAME VARCHAR2(45)
LAST_NAME VARCHAR2(45)
LAST_UPDATE TIMESTAMP
FILM
FILM_ID NUMBER(22) PK
TITLE VARCHAR2(255)
DESCRIPTION CLOB N
RELEASE_YEAR VARCHAR2(4) N
LANGUAGE_ID NUMBER(22) FK
ORIGINAL_LANGUAGE_ID NUMBER(22) N FK
RENTAL_DURATION NUMBER(22)
RENTAL_RATE NUMBER(4,2)
LENGTH NUMBER(22) N
REPLACEMENT_COST NUMBER(5,2)
RATING VARCHAR2(10) N
SPECIAL_FEATURES VARCHAR2(100) N
LAST_UPDATE TIMESTAMP
FILM_ACTOR
ACTOR_ID NUMBER(22) PK FK
FILM_ID NUMBER(22) PK FK
LAST_UPDATE TIMESTAMP
CATEGORY
CATEGORY_ID NUMBER(22) PK
NAME VARCHAR2(25)
LAST_UPDATE TIMESTAMP
FILM_CATEGORY
FILM_ID NUMBER(22) PK FK
CATEGORY_ID NUMBER(22) PK FK
LAST_UPDATE TIMESTAMP

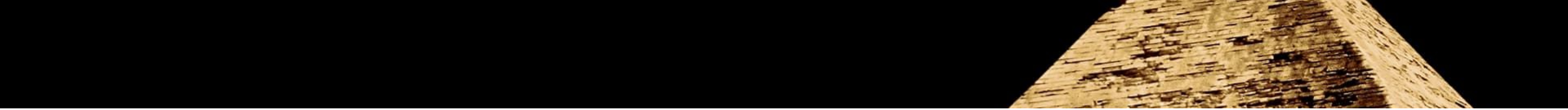

STAFF
STAFF_ID NUMBER(22) PK
FIRST_NAME VARCHAR2(45)
LAST_NAME VARCHAR2(45)
ADDRESS_ID NUMBER(22) FK
PICTURE BLOB N
EMAIL VARCHAR2(50) N
STORE_ID NUMBER(22) FK
ACTIVE NUMBER(22)
USERNAME VARCHAR2(16)
PASSWORD VARCHAR2(40) N
LAST_UPDATE TIMESTAMP
PAYMENT
PAYMENT_ID NUMBER(22) PK
CUSTOMER_ID NUMBER(22) FK
STAFF_ID NUMBER(22) FK
RENTAL_ID NUMBER(22) N FK
AMOUNT NUMBER(5,2)
PAYMENT_DATE TIMESTAMP
LAST_UPDATE TIMESTAMP
INVENTORY
INVENTORY_ID NUMBER(22) PK
FILM_ID NUMBER(22) FK
STORE_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
RENTAL
RENTAL_ID NUMBER(22) PK
RENTAL_DATE TIMESTAMP
INVENTORY_ID NUMBER(22) FK
CUSTOMER_ID NUMBER(22) FK
RETURN_DATE TIMESTAMP N
STAFF_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
CUSTOMER
CUSTOMER_ID NUMBER(22) PK
STORE_ID NUMBER(22) FK
FIRST_NAME VARCHAR2(45)
LAST_NAME VARCHAR2(45)
EMAIL VARCHAR2(50) N
ADDRESS_ID NUMBER(22) FK
ACTIVE CHAR(1)
CREATE_DATE DATE
LAST_UPDATE TIMESTAMP
STORE
STORE_ID NUMBER(22) PK
MANAGER_STAFF_ID NUMBER(22) FK
ADDRESS_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
ADDRESS
ADDRESS_ID NUMBER(22) PK
ADDRESS VARCHAR2(50)
ADDRESS2 VARCHAR2(50) N
DISTRICT VARCHAR2(20)
CITY_ID NUMBER(22) FK
POSTAL_CODE VARCHAR2(10) N
PHONE VARCHAR2(20)
LAST_UPDATE TIMESTAMP
CITY
CITY_ID NUMBER(22) PK
CITY VARCHAR2(50)
COUNTRY_ID NUMBER(22) FK
LAST_UPDATE TIMESTAMP
COUNTRY
COUNTRY_ID NUMBER(22) PK
COUNTRY VARCHAR2(50)
LAST_UPDATE TIMESTAMP N

# Common Table Expression

```sql
select
    c.first_name || ' ' || c.last_name,
    sum(amount)
from
    customer c
        join (
            select r.customer_id
            from rental r
                inner join inventory i on r.inventory_id = i.inventory_id
                inner join film f on i.film_id = f.film_id
                inner join film_category fc on f.film_id = fc.film_id
                inner join category cat on cat.category_id = fc.category_id
            where cat.name = 'Documentary'
            group by r.customer_id
        ) r on c.customer_id = r.customer_id
        join payment ps on c.customer_id = ps.customer_id
group by c.first_name, c.last_name
order by 2 desc;
```



```
var result = foo(bar(baz(1, foo(2,3)), really(4, tooMuch(5,6))), 7);
```



```
var first = tooMuch(5, 6);
var second = foo(2, 3);
var third = really(4, first);
var fourth = baz(1, second);
var last = bar(fourth, third);
var result = foo(last, 7);
```


Better readability

# Better readability

```
with documentary_rentals as (
    select r.customer_id
    from rental r
             inner join inventory i on r.inventory_id = i.inventory_id
             inner join film f on i.film_id = f.film_id
             inner join film_category fc on f.film_id = fc.film_id
             inner join category cat on cat.category_id = fc.category_id
    where cat.name = 'Documentary'
    group by r.customer_id
)
select
    c.first_name || ' ' || c.last_name,
    sum(ps.amount)
from
    customer c
        join documentary_rentals r on c.customer_id = r.customer_id
        join payment ps on c.customer_id = ps.customer_id
group by c.first_name, c.last_name
order by 2 desc
```

Naming
is **important**

# Naming is **important**

```sql
with documentary_rentals as (
    select r.customer_id
    from rental r
             inner join inventory i on r.inventory_id = i.inventory_id
             inner join film f on i.film_id = f.film_id
             inner join film_category fc on f.film_id = fc.film_id
             inner join category cat on cat.category_id = fc.category_id
    where cat.name = 'Documentary'
    group by r.customer_id
)
select
    c.first_name || ' ' || c.last_name,
    sum(ps.amount)
from
    customer c
        join documentary_rentals r on c.customer_id = r.customer_id
        join payment ps on c.customer_id = ps.customer_id
group by c.first_name, c.last_name
order by 2 desc
```

# Faster Execution

# Faster Execution

```
with
  first_cte as (…),
  second_cte as (…),
  third_cte as (…)
select
   c.first_name || ' ' || c.last_name,
   sum(ps.amount)
from
   …
group by c.first_name, c.last_name
order by 2 desc
```


Window Functions

From
Join

Where

Group by

Aggregate
Functions

Having

**Window
Functions**

Select

Distinct

Union
Intersect
Except

Order by

Offset

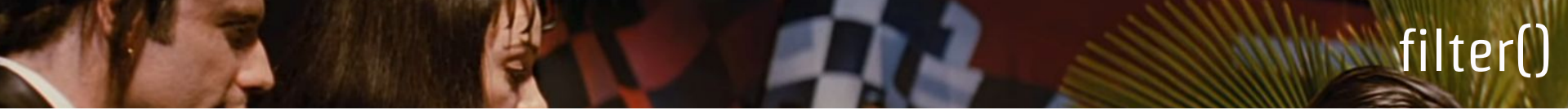
Limit
Fetch
Top

# Window Functions

```sql
select rating, count(*) as "number of movie"
from film
group by rating;
```

| rating | number of movie |
|---|---|
| PG | 194 |
| G | 178 |
| NC-17 | 210 |
| PG-13 | 223 |
| R | 195 |

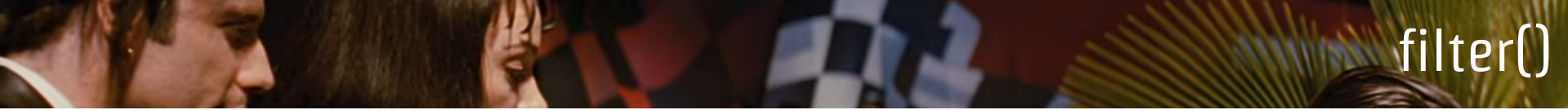

filter()



# filter()

```sql
select
    count(*) as "all",
    count(*) filter ( where rating = 'PG' ) as "PG",
    count(*) filter ( where rating = 'G' ) as "G",
    count(*) filter ( where rating = 'NC-17' ) as "NC-17",
    count(*) filter ( where rating = 'PG-13' ) as "PG-13",
    count(*) filter ( where rating = 'R' ) as "R"
from film;
```

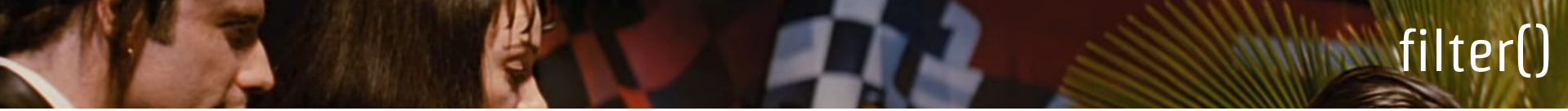
```
 all  | PG  |  G  | NC-17 | PG-13 |  R
------+-----+-----+-------+-------+-----
 1000 | 194 | 178 |   210 |   223 | 195
```



# filter()

```
select

   count(*) filter ( where length > 30 ) as "> 30m",
   count(*) filter ( where length > 60 ) as "> 60m",
   count(*) filter ( where length > 90 ) as "> 90m",
   count(*) filter ( where length > 120 ) as "> 120m",
   count(*) filter ( where length > 180 ) as "> 180m"
from film;
```

```
 > 30m | > 60m | > 90m | > 120m | > 180m
-------+-------+-------+--------+--------
  1000 |   896 |   675 |    457 |     39
```



# filter()

```
select
   count(*) as "all",
   count(*) filter ( where rating = 'PG' ) as "PG",
   count(*) filter ( where rating = 'G' ) as "G",
   count(*) filter ( where rating = 'NC-17' ) as "NC-17",
   count(*) filter ( where rating = 'PG-13' ) as "PG-13",
   count(*) filter ( where rating = 'R' ) as "R",
   count(*) filter ( where length > 30 ) as "> 30m",
   count(*) filter ( where length > 60 ) as "> 60m",
   count(*) filter ( where length > 90 ) as "> 90m",
   count(*) filter ( where length > 120 ) as "> 120m",
   count(*) filter ( where length > 180 ) as "> 180m"
from film
```

| all | PG | G | NC-17 | PG-13 | R | > 30m | > 60m | > 90m | > 120m | > 180m |
|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | 194 | 178 | 210 | 223 | 195 | 1000 | 896 | 675 | 457 | 39 |

# LAG() & LEAD()

# LAG() & LEAD()

```
select count(*) as "# of time rented", title
from film
   inner join inventory i using (film_id)
   inner join rental r using(inventory_id)
   inner join customer c using(customer_id)
group by film_id, title
order by "# of time rented" desc, film_id;
```

| # of time rented | title |
|---|---|
| **34** | **BUCKET BROTHERHOOD** |
| 33 | ROCKETEER MOTHER |
| 32 | FORWARD TEMPLE |
| 32 | GRIT CLOCKWORK |
| 32 | JUGGLER HARDLY |
| … | … |

# LAG() & LEAD()

```
select
 c.first_name || ' ' || c.last_name as "rented by", date_trunc('day', r.rental_date),

 lag(c.first_name || ' ' || c.last_name)
  over (partition by i.inventory_id order by rental_date) as "rented previously by",
 lead(c.first_name || ' ' || c.last_name)
  over (partition by i.inventory_id order by rental_date) as "rented later by"
from film
 inner join inventory i using (film_id)
 inner join rental r using(inventory_id)
 inner join customer c using(customer_id)

where title = 'BUCKET BROTHERHOOD';
```

| rented by | date_trunc | rented previously by | rented later by |
|---|---|---|---|
| **LORETTA CARPENTER** | 2005-07-11 | <null> | **WILLIE MARKHAM** |
| **WILLIE MARKHAM** | 2005-07-28 | **LORETTA CARPENTER** | **BOBBY BOUDREAU** |
| **BOBBY BOUDREAU** | 2005-08-21 | **WILLIE MARKHAM** | <null> |
| MONICA HICKS | 2005-05-25 | <null> | ANTONIO MEEK |
| ANTONIO MEEK | 2005-06-17 | MONICA HICKS | SETH HANNON |
| SETH HANNON | 2005-07-12 | ANTONIO MEEK | TARA RYAN |
| TARA RYAN | 2005-07-29 | SETH HANNON | JUAN FRALEY |
| JUAN FRALEY | 2005-08-17 | TARA RYAN | <null> |

# LAG() & LEAD()

```
select
 c.first_name || ' ' || c.last_name as "rented by", date_trunc('day', r.rental_date),

 lag(c.first_name || ' ' || c.last_name) over rd as "rented previously by",

 lead(c.first_name || ' ' || c.last_name) over rd as "rented later by"

from film
 inner join inventory i using (film_id)
 inner join rental r using(inventory_id)
 inner join customer c using(customer_id)

where title = 'BUCKET BROTHERHOOD'
window rd as (partition by i.inventory_id order by rental_date);
```

| rented by | date_trunc | rented previously by | rented later by |
|---|---|---|---|
| **LORETTA CARPENTER** | 2005-07-11 | <null> | **WILLIE MARKHAM** |
| **WILLIE MARKHAM** | 2005-07-28 | **LORETTA CARPENTER** | **BOBBY BOUDREAU** |
| **BOBBY BOUDREAU** | 2005-08-21 | **WILLIE MARKHAM** | <null> |
| MONICA HICKS | 2005-05-25 | <null> | ANTONIO MEEK |
| ANTONIO MEEK | 2005-06-17 | MONICA HICKS | SETH HANNON |
| SETH HANNON | 2005-07-12 | ANTONIO MEEK | TARA RYAN |
| TARA RYAN | 2005-07-29 | SETH HANNON | JUAN FRALEY |
| JUAN FRALEY | 2005-08-17 | TARA RYAN | <null> |

# [dense_]rank()

# [dense_]rank()

```
select f.title, count(*) as "# of time rented"
from film f
 inner join inventory i on f.film_id = i.film_id
 inner join rental r on i.inventory_id = r.inventory_id
group by f.film_id, f.title
order by 2 desc
```

| title | # of time rented |
|---|---|
| BUCKET BROTHERHOOD | 34 |
| ROCKETEER MOTHER | 33 |
| FORWARD TEMPLE | 32 |
| GRIT CLOCKWORK | 32 |
| JUGGLER HARDLY | 32 |
| RIDGEMONT SUBMARINE | 32 |
| SCALAWAG DUCK | 32 |
| APACHE DIVINE | 31 |
| GOODFELLAS SALUTE | 31 |
| … | … |

# [dense_]rank()

```
with film_order_by_rent_success(title, times_rented) as (
 select f.title, count(*) as "# of time rented"
 from film f
  inner join inventory i on f.film_id = i.film_id
  inner join rental r on i.inventory_id = r.inventory_id
 group by f.film_id, f.title
)
select
 title, times_rented,
 rank() over (order by times_rented desc),
 dense_rank() over (order by times_rented desc)
from film_order_by_rent_success;
```

| title | times_rented | rank | dense_rank |
|---|---|---|---|
| BUCKET BROTHERHOOD | 34 | 1 | 1 |
| ROCKETEER MOTHER | 33 | 2 | 2 |
| FORWARD TEMPLE | 32 | 3 | 3 |
| GRIT CLOCKWORK | 32 | 3 | 3 |
| JUGGLER HARDLY | 32 | 3 | 3 |
| RIDGEMONT SUBMARINE | 32 | 3 | 3 |
| SCALAWAG DUCK | 32 | 3 | 3 |
| APACHE DIVINE | 31 | 8 | 4 |
| GOODFELLAS SALUTE | 31 | 8 | 4 |
| … | | | |

# [dense_]rank()

```
with
  film_order_by_rent_success(title, times_rented) as (…),
  film_ranked (title, times_ranted, rank, dense_rank) as (
    select
      title, times_rented,
      rank() over (order by times_rented desc),
      dense_rank() over (order by times_rented desc)
    from film_order_by_rent_success
)
select title, times_ranted, rank, dense_rank
from film_ranked
limit 4
```

| title | times_rented | rank | dense_rank |
|---|---|---|---|
| **BUCKET BROTHERHOOD** | **34** | **1** | **1** |
| **ROCKETEER MOTHER** | **33** | **2** | **2** |
| **FORWARD TEMPLE** | **32** | **3** | **3** |
| **GRIT CLOCKWORK** | **32** | **3** | **3** |
| JUGGLER HARDLY | 32 | 3 | 3 |
| RIDGEMONT SUBMARINE | 32 | 3 | 3 |
| SCALAWAG DUCK | 32 | 3 | 3 |
| APACHE DIVINE | 31 | 8 | 4 |
| GOODFELLAS SALUTE | 31 | 8 | 4 |

…

# [dense_]rank()

```
with
  film_order_by_rent_success(title, times_rented) as (…),
  film_ranked (title, times_ranted, rank, dense_rank) as (
    select
      title, times_rented,
      rank() over (order by times_rented desc),
      dense_rank() over (order by times_rented desc)
    from film_order_by_rent_success
)
select title, times_ranted, rank, dense_rank
from film_ranked
where rank ⩽ 4
```

| title | times_rented | rank | dense_rank |
|---|---|---|---|
| BUCKET BROTHERHOOD | 34 | 1 | 1 |
| ROCKETEER MOTHER | 33 | 2 | 2 |
| FORWARD TEMPLE | 32 | 3 | 3 |
| GRIT CLOCKWORK | 32 | 3 | 3 |
| JUGGLER HARDLY | 32 | 3 | 3 |
| RIDGEMONT SUBMARINE | 32 | 3 | 3 |
| SCALAWAG DUCK | 32 | 3 | 3 |
| APACHE DIVINE | 31 | 8 | 4 |
| GOODFELLAS SALUTE | 31 | 8 | 4 |

…

# [dense_]rank()

```
with
  film_order_by_rent_success(title, times_rented) as (…),
  film_ranked (title, times_ranted, rank, dense_rank) as (
    select
      title, times_rented,
      rank() over (order by times_rented desc),
      dense_rank() over (order by times_rented desc)
    from film_order_by_rent_success
)
select title, times_ranted, rank, dense_rank
from film_ranked
where dense_rank ⩽ 4
```

| title | times_rented | rank | dense_rank |
|---|---|---|---|
| BUCKET BROTHERHOOD | 34 | 1 | 1 |
| ROCKETEER MOTHER | 33 | 2 | 2 |
| FORWARD TEMPLE | 32 | 3 | 3 |
| GRIT CLOCKWORK | 32 | 3 | 3 |
| JUGGLER HARDLY | 32 | 3 | 3 |
| RIDGEMONT SUBMARINE | 32 | 3 | 3 |
| SCALAWAG DUCK | 32 | 3 | 3 |
| APACHE DIVINE | 31 | 8 | 4 |
| GOODFELLAS SALUTE | 31 | 8 | 4 |
| … | | | |

# row_number()

# row_number()

```
create table film_with_duplicate (
   film_id serial primary key,
   title character varying(255) not null
);


insert into film_with_duplicate (title)
   -- all original films are inserted
   (select title from film)
union all
   -- and we insert 40 random films from the same list
   (select title from film order by random() limit 40);
```

# row_number()

```
select count(*) as "number of films"
from film_with_duplicate;
```

| number of films |
| --- |
| 1040 |

# row_number()

```
select title, film_id
from film_with_duplicate
order by title;
```

| title | film_id |
|---|---|
| ACADEMY DINOSAUR | 1074 |
| ACE GOLDFINGER | 1747 |
| **ADAPTATION HOLES** | **1986** |
| **ADAPTATION HOLES** | **2034** |
| AFFAIR PREJUDICE | 1614 |
| AFRICAN EGG | 1047 |

# row_number()

```
select title, film_id
       row_number() over (partition by title) as duplication_times,
from film_with_duplicate
```

| title | duplication_times | film_id |
|---|---|---|
| ACADEMY DINOSAUR | 1 | 1074 |
| ACE GOLDFINGER | 1 | 1747 |
| ADAPTATION HOLES | 1 | 1986 |
| **ADAPTATION HOLES** | **2** | **2034** |
| AFFAIR PREJUDICE | 1 | 1614 |
| AFRICAN EGG | 1 | 1047 |

# row_number()

```
with iteration_of_each_film_entry as (
    select title, film_id,
           row_number() over (partition by title) as duplication_times
    from film_with_duplicate
)
delete from film_with_duplicate
where film_id IN (
    select distinct film_id
    from iteration_of_each_film_entry
    where duplication_times > 1
);
```

| title | duplication_times | film_id |
|---|---|---|
| ACADEMY DINOSAUR | 1 | 1074 |
| ACE GOLDFINGER | 1 | 1747 |
| ADAPTATION HOLES | 1 | 1986 |
| **ADAPTATION HOLES** | **2** | **2034** |
| AFFAIR PREJUDICE | 1 | 1614 |
| AFRICAN EGG | 1 | 1047 |

# row_number()

```
40 rows affected in 18 ms
```


Join Lateral

# Join Lateral

```
select f.title, rental_date,
  row_number() over (partition by f.title order by rental_date desc)
from film f
join inventory i on f.film_id = i.film_id
join rental r on i.inventory_id = r.inventory_id
order by title, rental_date desc
```

| title | rental_date | row_number |
|---|---|---|
| **ACADEMY DINOSAUR** | **2005-08-23 01:01:01** | **1** |
| **ACADEMY DINOSAUR** | **2005-08-22 23:56:37** | **2** |
| **ACADEMY DINOSAUR** | **2005-08-22 00:44:08** | **3** |
| ACADEMY DINOSAUR | 2005-08-21 21:27:43 | 4 |
| ACADEMY DINOSAUR | 2005-08-21 18:32:42 | 5 |
| ACADEMY DINOSAUR | 2005-08-21 00:30:32 | 6 |
| ACADEMY DINOSAUR | 2005-08-18 18:36:16 | 7 |
| ACADEMY DINOSAUR | 2005-08-02 20:13:10 | 8 |
| … | | |

# Join Lateral

```sql
with all_rentals (film, rental_date, iteration) as (
   select f.title, rental_date,
     row_number() over (partition by f.title order by rental_date desc)
   from film f
   join inventory i on f.film_id = i.film_id
   join rental r on i.inventory_id = r.inventory_id
   order by title, rental_date desc
)
select film, rental_date
from all_rentals
where iteration ⩽ 3;
```

| film | rental_date |
|---|---|
| **ACADEMY DINOSAUR** | **2005-08-23 01:01:01** |
| **ACADEMY DINOSAUR** | **2005-08-22 23:56:37** |
| **ACADEMY DINOSAUR** | **2005-08-22 00:44:08** |
| ACE GOLDFINGER | 2006-02-14 15:16:03 |
| ACE GOLDFINGER | 2005-08-22 16:59:05 |
| ACE GOLDFINGER | 2005-08-17 09:33:02 |
| ADAPTATION HOLES | 2005-08-23 13:54:39 |
| ADAPTATION HOLES | 2005-08-22 03:52:21 |
| … | |


Do you know

# for-each

?

# Join Lateral

```
select film.title, recently_rented.rental_date
from film
```

```
            title
-----------------------------
 ACADEMY DINOSAUR
 ACE GOLDFINGER
 ADAPTATION HOLES
 AFFAIR PREJUDICE
 AFRICAN EGG
 AGENT TRUMAN
 AIRPLANE SIERRA
 AIRPORT POLLOCK
…
```

# Join Lateral

```sql
select film.title, recently_rented.rental_date
from film
left join lateral (
   select *
   from inventory i
   join rental r on i.inventory_id = r.inventory_id
   where i.film_id = film.film_id -- film from outer query
   order by rental_date desc
   limit 3
) as recently_rented on film.film_id = recently_rented.film_id
order by film.title;
```

```
            title             |     rental_date
------------------------------+---------------------
 ACADEMY DINOSAUR             | 2005-08-23 01:01:01
 ACADEMY DINOSAUR             | 2005-08-22 23:56:37
 ACADEMY DINOSAUR             | 2005-08-22 00:44:08
 ACE GOLDFINGER               | 2006-02-14 15:16:03
 ACE GOLDFINGER               | 2005-08-22 16:59:05
 ACE GOLDFINGER               | 2005-08-17 09:33:02
 ADAPTATION HOLES             | 2005-08-23 13:54:39
 ADAPTATION HOLES             | 2005-08-22 03:52:21
…
```

# Pagination

# Pagination

```sql
select
   f.title as film,
   c.first_name || ' ' || c.last_name as "rented by",
   r.rental_date as from,
   r.return_date as to,
from rental r
  inner join inventory i using(inventory_id)
  inner join film f using(film_id)
  inner join customer c using(customer_id)
order by r.rental_date, r.return_date desc
limit 5 offset 5*3;
```

| film | rented by | from | to |
|---|---|---|---|
| BOOGIE AMELIE | STEVEN CURLEY | 2005-05-25 00:43:11 | 2005-05-26 04:42:11 |
| CONTACT ANONYMOUS | ISAAC OGLESBY | 2005-05-25 01:06:36 | 2005-05-27 00:43:36 |
| ROMAN PUNK | RUTH MARTINEZ | 2005-05-25 01:10:47 | 2005-05-31 06:35:47 |
| HOLLOW JEOPARDY | RONNIE RICKETTS | 2005-05-25 01:17:24 | 2005-05-31 06:00:24 |
| **SCISSORHANDS SLUMS** | **ROBERTA HARPER** | **2005-05-25 01:48:41** | **2005-05-27 02:20:41** |

# Pagination

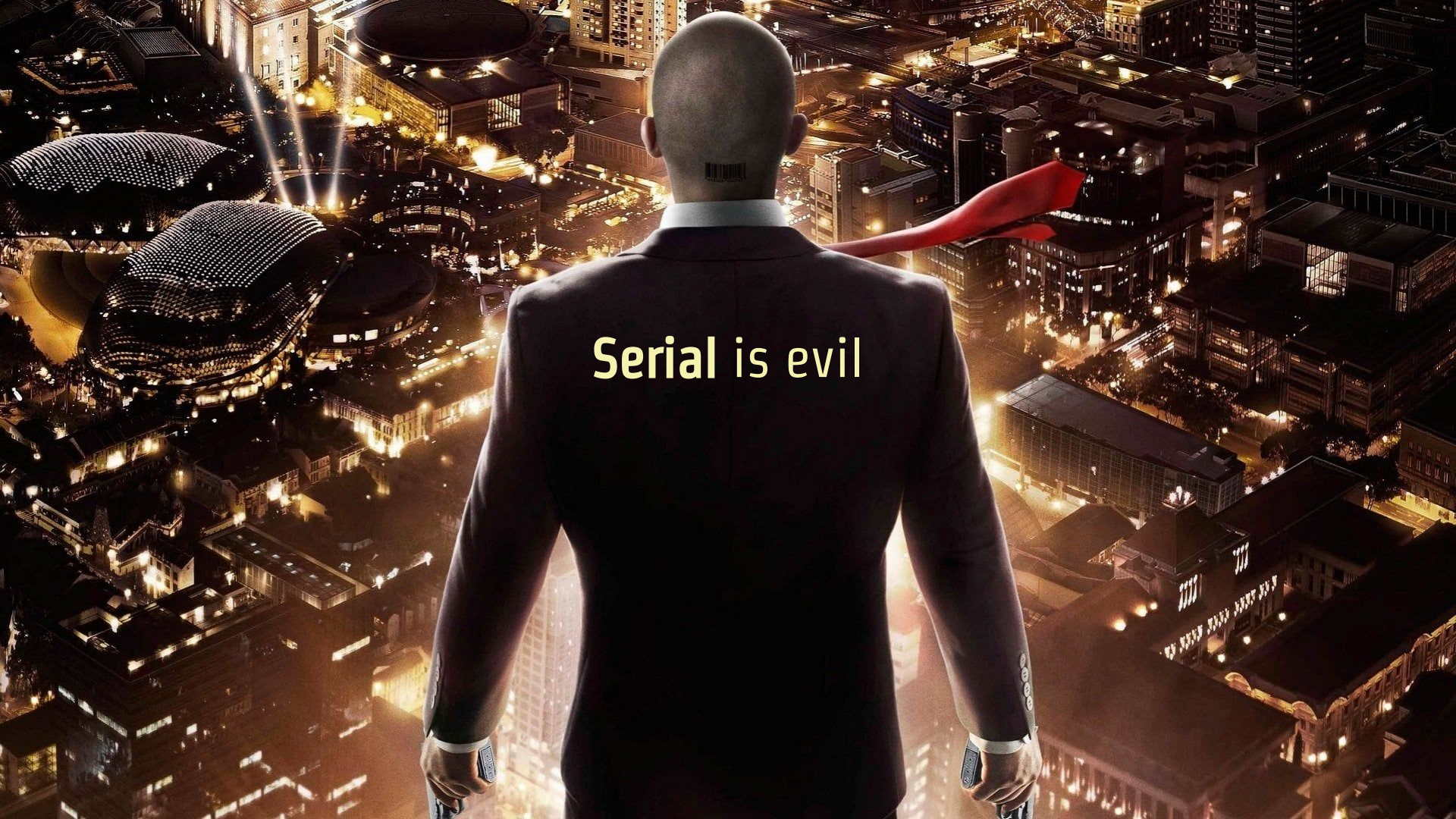
```sql
select
   f.title as film,
   c.first_name || ' ' || c.last_name as "rented by",
   r.rental_date as from,
   r.return_date as to,
from rental r
  inner join inventory i using(inventory_id)
  inner join film f using(film_id)
  inner join customer c using(customer_id)
order by r.rental_date, r.return_date desc
limit 5 offset 5*4;
```

| film | rented by | from | to |
|---|---|---|---|
| **APACHE DIVINE** | **CRAIG MORRELL** | **2005-05-25 01:59:46** | **2005-05-26 01:01:46** |
| CLOSER BANG | RAUL FORTIER | 2005-05-25 02:19:23 | 2005-05-26 04:52:23 |
| WHALE BIKINI | BARRY LOVELACE | 2005-05-25 02:40:21 | 2005-05-29 06:34:21 |
| REDS POCUS | JUAN FRALEY | 2005-05-25 02:53:02 | 2005-05-27 01:15:02 |
| SUN CONFESSIONS | PAMELA BAKER | 2005-05-25 03:21:20 | 2005-05-27 21:25:20 |

# Pagination

```sql
--- Last element of Page 3:
---   r.rental_date = '2005-05-25 01:48:41'
---   r.return_date = '2005-05-27 02:20:41'
select
   f.title as film,
   c.first_name || ' ' || c.last_name as "rented by",
   r.rental_date as "from",
   r.return_date as "to"
from rental r
 inner join inventory i using(inventory_id)
 inner join film f using(film_id)
 inner join customer c using(customer_id)
where (r.rental_date, r.return_date) > ('2005-05-25 01:48:41'::timestamp, '2005-05-27 02:20:41'::timestamp)
-- where   r.rental_date > '2005-05-25 01:48:41.000000'::timestamp
-- and     r.last_update > '2005-05-27 02:20:41.000000'::timestamp
order by r.rental_date, r.return_date desc
limit 5;
```

| film | rented by | from | to |
|---|---|---|---|
| APACHE DIVINE | CRAIG MORRELL | 2005-05-25 01:59:46 | 2005-05-26 01:01:46 |
| CLOSER BANG | RAUL FORTIER | 2005-05-25 02:19:23 | 2005-05-26 04:52:23 |
| WHALE BIKINI | BARRY LOVELACE | 2005-05-25 02:40:21 | 2005-05-29 06:34:21 |
| REDS POCUS | JUAN FRALEY | 2005-05-25 02:53:02 | 2005-05-27 01:15:02 |
| SUN CONFESSIONS | PAMELA BAKER | 2005-05-25 03:21:20 | 2005-05-27 21:25:20 |


Serial is evil


```sql
create table film_serial (
    id serial primary key,
    title character varying(255) not null
);


insert into film_serial (title)
values
    ('Matrix'),
    ('Matrix Reloaded'),
    ('Matrix Revolutions'),
    ('Matrix Resurrections')
returning id, title;
```


```
id |         title
---+-----------------------
 1 | Matrix
 2 | Matrix Reloaded
 3 | Matrix Revolutions
 4 | Matrix Resurrections
```

```
select currval('film_serial_id_seq'::regclass);
```

UUID is better!

# UUID is better!

```
create extension "uuid-ossp";
create table if not exists film_uuid (
   id uuid primary key default uuid_generate_v4(),
   title character varying(255) not null
);
```

```
insert into film_uuid (title)
values ('Matrix'),
       ('Matrix Reloaded'),
       ('Matrix Revolutions'),
       ('Matrix Resurrections')
returning id, title;
```

# UUID is better!

```
                  id                  |        title
--------------------------------------+----------------------
60b8a533-2680-4eaa-a75e-128aa0c67df9 | Matrix
45d3f3c9-cbbe-450f-8d14-cbdfd30c1109 | Matrix Reloaded
d1687d3e-3d75-4620-8ade-f26c6e8fd8ae | Matrix Revolutions
16f8daf5-2bd5-4816-ac6d-79042ab3b584 | Matrix Resurrections
```

# UUID is better!

```
insert into film_uuid (id, title)
values
  ('4339890d-706f-4950-a862-542f288dca2e'::uuid, 'Matrix'),
  ('b4774c08-8be9-44a4-90aa-8b9af26a5c13'::uuid, 'Matrix Reloaded'),
  ('3f09a064-2f32-41b9-be24-94cc0b068a40'::uuid, 'Matrix Revolutions'),
  ('fae647ab-7c60-4e7c-b4a5-2188b3845f83'::uuid, 'Matrix Resurrections')
-- returning id, title;
```


Constraints

# Constraints

```
create table film_with_isan (
 film_id serial primary key,
 title character varying(255) not null,
 isan character varying(255)




);
```

# Constraints

```
create table film_with_isan (
 film_id serial primary key,
 title character varying(255) not null,
 isan character varying(255),

 constraint has_valid_isan check (
  isan is null or
  isan ~* 'ISAN [0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-z]-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-z]'
    --       ISAN     0000   -    0001   -    8947   -    0000   -       8-    0000   -    0000   -       D
 )

);
```

# Constraints

```
create table film_with_isan (
 film_id serial primary key,
 title character varying(255) not null,
 isan character varying(255),

 constraint has_valid_isan check (
  isan is null or
  isan ~* 'ISAN [0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-z]-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-z]'
   --       ISAN     0000   -    0001   -    8947   -    0000   -       8-    0000   -    0000   -       D
 ),
 constraint isan_is_unique unique (isan)
);
```

# Constraints

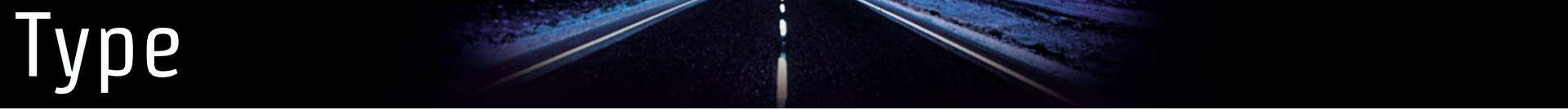
```
sakila.public> insert into film_with_isan (title, isan)
               values ('INVALID_ISAN','63998367-c0c9-4fe0-82fc-ba4c96890ee1');

[2023-03-25 17:04:37] [23514] ERROR: new row for relation "film_with_isan" violates check constraint "has_valid_isan"
[2023-03-25 17:04:37] Detail: Failing row contains (2042, INVALID_ISAN, 63998367-c0c9-4fe0-82fc-ba4c96890ee1).
```

# Constraints

```
sakila.public> insert into film_with_isan (title, isan)
               values ('REAL_ISAN', 'ISAN 0000-0001-68EC-0000-X-0000-0000-C');

[2023-03-25 17:07:47] 1 row affected in 13 ms

sakila.public> insert into film_with_isan (title, isan)
               values ('REAL_ISAN', 'ISAN 0000-0001-68EC-0000-X-0000-0000-C');

[2023-03-25 17:08:21] [23505] ERROR: duplicate key value violates unique constraint "isan_is_unique"
[2023-03-25 17:08:21] Detail: Key (isan)=(ISAN 0000-0001-68EC-0000-X-0000-0000-C) already exists.
```

# Type

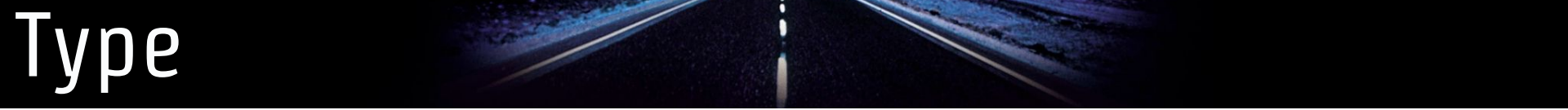
# Type



```
create type isan_type as (
   block_1 varchar(4), block_2 varchar(4), block_3 varchar(4), block_4 varchar(4),
   block_5 varchar(1), block_6 varchar(4), block_7 varchar(4), block_8 varchar(1)
);
```

```
create domain isan as isan_type check (
       (value).block_1 is not null and (value).block_1 ~* '[0-9a-f]{4}'
   and (value).block_2 is not null and (value).block_2 ~* '[0-9a-f]{4}'
   and (value).block_3 is not null and (value).block_3 ~* '[0-9a-f]{4}'
   and (value).block_4 is not null and (value).block_4 ~* '[0-9a-f]{4}'
   and (value).block_5 is not null and (value).block_5 ~* '[0-9a-f]'
   and (value).block_6 is not null and (value).block_6 ~* '[0-9a-f]{4}'
   and (value).block_7 is not null and (value).block_7 ~* '[0-9a-f]{4}'
   and (value).block_8 is not null and (value).block_8 ~* '[0-9a-f]'
);
```

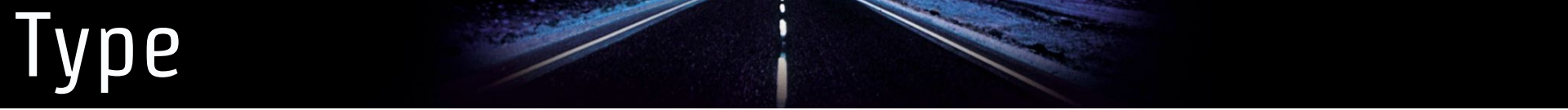
# Type



```
select ('0000', '0001', '68ec', '0000', 'a', '0000', '0000', 'c')::isan;
```

```
                isan
-------------------------------------
(0000,0001,68ec,0000,a,0000,0000,c)
```

# Type



```
select (null, '0001', '68ec', '0000', 'a', '0000', '0000', 'c')::isan;
```

```
[2023-03-29 21:15:46] [23514] ERROR: value for domain isan violates check
constraint "isan_check"
```

# Type

```
create function isan_to_text(isan) … $$
create function text_to_isan(isan) … $$
create function isan_exception(text) … $$
create function isan_equals(isan) … $$
create function isan_not_equals(isan) … $$
```

```
select
   isan_equals(
     text_to_isan('ISAN 0000-0001-68EC-0000-A-0000-0000-C'),
     text_to_isan('ISAN 0000-0001-68EC-0000-A-0000-0000-C')
   ) as isan_equality,

   isan_not_equals(
       ('0000', '0001', '68ec', '0000', 'a', '0000', '0000', 'c')::isan,
       text_to_isan('ISAN 0000-0001-68EC-0000-A-0000-0000-C')
   ) as isan_different
;
```

# Type


```
create operator = (
 leftarg = isan,
 rightarg = isan,
 procedure = isan_equals,
 commutator = =,
 negator = ≠,
 hashes,
 merges
);
```

```
create operator ≠ (
 leftarg = isan,
 rightarg = isan,
 function = isan_not_equals,
 commutator = ≠,
 negator = =,
 hashes,
 merges
);
```

```
select
  text_to_isan('ISAN 0000-0001-68EC-0000-A-0000-0000-C') = text_to_isan('ISAN 0000-0001-68EC-0000-A-0000-0000-C'),
  text_to_isan('ISAN 0000-0001-68EC-0000-A-0000-0000-C') ≠ text_to_isan('ISAN 0000-0001-68EC-0000-A-0000-0000-C');
```

# on conflict...


UNDER EIGHTEEN
ALL VALLEY KARATE CHAMPIONSHIP
SEMI-FINALS
SMITH
VIDAL
LAWRENCE
ESPINOZA
HALL
BROWN
VIDAL
LAWRENCE
BROWN
LABUSSO
FINALS
LAWRENCE
REFEREE
OFFICIAL


# on conflict

```
create table film (
   film_id serial primary key,
   title character varying(255) not null,

   last_update timestamp with time zone default now() not null,
   constraint unique_title unique (title)
);
```


# on conflict

```
select title, last_update
from film
```

```
     title       |          last_update
-----------------+-------------------------------
ACADEMY DINOSAUR | 2000-01-01 20:00:01.000000+00
ACE GOLDFINGER   | 2000-01-01 20:00:01.000000+00
```


# on conflict

```
insert into film (title, last_update)
values ('ACADEMY DINOSAUR', '2023-01-02 03:04:05.678910+00'::timestamptz),
       ('ACE GOLDFINGER', '2023-01-02 03:04:05.678910+00'::timestamptz),
       ('JUST FOR DEMO', '2023-01-02 03:04:05.678910+00'::timestamptz)
on conflict on constraint unique_title do update
set last_update = now();
```

```
     title       |          last_update
-----------------+-------------------------------
ACADEMY DINOSAUR | 2023-06-28 20:20:20.000000+00
ACE GOLDFINGER   | 2023-06-28 20:20:20.000000+00
JUST FOR DEMO    | 2023-01-02 03:04:05.000000+00
```


Merge


```
create table film (
    film_id serial primary key,
    title character varying(255) not null,

    rating mpaa_rating default 'G'::mpaa_rating,
    number_of_time_requested integer default 0
);

create table requested_film (
    film_id serial primary key,
    title character varying(255) not null,
    rating mpaa_rating default 'G'::mpaa_rating
);
```

# Merge

```
insert into requested_film (title, rating)
values
   ('ACADEMY DINOSAUR', 'PG'),
   ('ACE GOLDFINGER', 'G'),
   ('JUST FOR MERGE DEMO', 'NC-17'),
   ('TOO VIOLENT MOVIE', 'R');
```

```
select title from requested_film order by title;
```

```
       title        | rating
--------------------+--------
ACADEMY DINOSAUR    | PG
ACE GOLDFINGER      | G
JUST FOR MERGE DEMO | NC-17
TOO VIOLENT MOVIE   | R
```

# Merge

```sql
select title, last_update
from film
where title in (
   'ACADEMY DINOSAUR',
   'ACE GOLDFINGER',
   'JUST FOR MERGE DEMO',
   'TOO VIOLENT MOVIE'
);
```

| title | number_of_time_requested |
|---|---|
| ACADEMY DINOSAUR | 1 |
| ACE GOLDFINGER | 2 |

# Merge

```
merge into film f
using requested_film rf
   on rf.title = f.title
when matched then
   update set number_of_time_requested = number_of_time_requested + 1
when not matched and rf.rating = 'R'::mpaa_rating then
   do nothing
when not matched then
   insert (title, rating, number_of_time_requested)
   values (rf.title, rf.rating, 1);
```


**requested_film**

| title | rating |
|---|---|
| ACADEMY DINOSAUR | PG |
| ACE GOLDFINGER | G |
| JUST FOR MERGE DEMO | NC-17 |
| TOO VIOLENT MOVIE | R |

```
merge into film f
using requested_film rf
   on rf.title = f.title
```

**film**

| title | number_of_time_requested |
|---|---|
| ACADEMY DINOSAUR | 1 |
| ACE GOLDFINGER | 2 |


```
merge into film f
using requested_film rf
   on rf.title = f.title
when matched then
   update set number_of_time_requested = number_of_time_requested + 1
when not matched and rf.rating = 'R'::mpaa_rating then
   do nothing
when not matched then
   insert (title, rating, number_of_time_requested)
   values (rf.title, rf.rating, 1);
```


```
select title, number_of_time_requested
from film
where title in (
   'ACADEMY DINOSAUR',
   'ACE GOLDFINGER',
   'JUST FOR MERGE DEMO',
   'TOO VIOLENT MOVIE'
)
```

| title | number_of_time_requested |
|---|---|
| ACADEMY DINOSAUR | 2 |
| ACE GOLDFINGER | 3 |
| JUST FOR MERGE DEMO | 1 |

# Listen | Notify

# Listen | Notify

```
create or replace function notify_new_film()
returns trigger
language plpgsql
as $$
declare
 channel text := TG_ARGV[0];
begin
 perform (
    with payload as (select NEW.film_id as id, NEW.title as title)
    select pg_notify(channel, row_to_json(payload)::text) from payload
 );
 return null;
end;
$$;
```

```
create trigger notify_clients
after insert on film_with_notification
for each row execute procedure notify_new_film('film.add');
```

# Listen | Notify

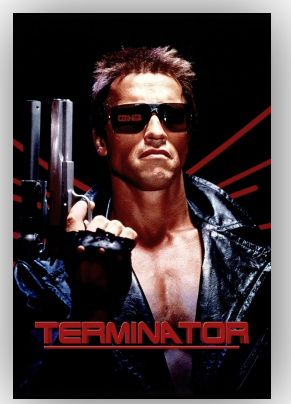
```
listen "film.add";
```

```
Asynchronous notification "film.add" with payload
"{"id":1010,"title":"Speed"}" received from server
process with PID 161.

Asynchronous notification "film.add" with payload
"{"id":1011,"title":"John Wick"}" received from server
process with PID 161.

Asynchronous notification "film.add" with payload
"{"id":1012,"title":"Point Break"}" received from server
process with PID 161.

Asynchronous notification "film.add" with payload
"{"id":1013,"title":"The Devil's Advocate"}" received
from server process with PID 161.

Asynchronous notification "film.add" with payload
"{"id":1014,"title":"The Lake House"}" received from
server process with PID 161.
```

```
insert into film_with_notification(title)
values
   ('Speed'), ('John Wick'), ('Point Break'),
   ('The Devil''s Advocate'), ('The Lake House');
```

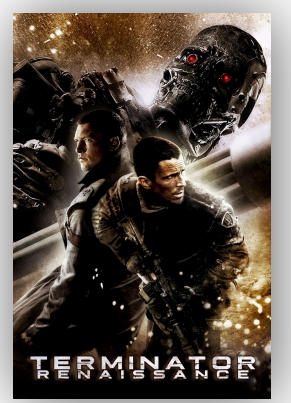
# ltree & more

# Itree & more


TERMINATOR


SCHWARZENEGGER
TERMINATOR 2
LE JUGEMENT DERNIER


TERMINATOR 3
LE SOULÈVEMENT DES MACHINES

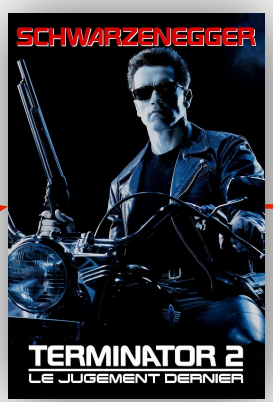

TERMINATOR
RENAISSANCE


TERMINATOR
GENISYS
LA MENACE A CHANGÉ.

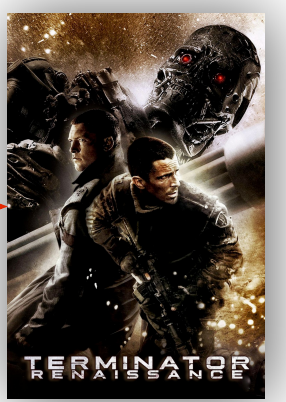

TERMINATOR
DARK FATE

# Itree & more


TERMINATOR


SCHWARZENEGGER
TERMINATOR 2
LE JUGEMENT DERNIER


TERMINATOR 3
LE SOULÈVEMENT DES MACHINES


TERMINATOR
RENAISSANCE


TERMINATOR
GENISYS
LA MENACE A CHANGÉ.


TERMINATOR
DARK FATE

# ltree & more

```sql
create extension ltree;

create table film_with_saga_order (
   film_id serial primary key,
   title character varying(255) not null,
   saga_order ltree
);



insert into film_with_saga_order (film_id, title, saga_order)
values
   (1, 'Terminator', '1'),
   (2, 'Terminator 2: Judgment Day', '1.2'),
   (3, 'Terminator 3: Rise of the Machines', '1.2.3'),
   (4, 'Terminator Salvation', '1.2.3.4'),
   (5, 'Terminator Genisys', '1.5'),
   (6, 'Terminator: Dark Fate', '1.2.6');
```

# ltree & more

```sql
with selected_movie as (
   select saga_order
   from film_with_saga_order
   where film_id = 6
)
select *
from film_with_saga_order
where saga_order @> (select saga_order from selected_movie);
-- @> means '[left ltree] is ancestor of [right ltree]
```

```
film_id |           title            | saga_order
--------+----------------------------+------------
      1 | Terminator                 | 1
      2 | Terminator 2: Judgment Day | 1.2
      6 | Terminator: Dark Fate      | 1.2.6
```

Itree & **more**

# ltree & more

```
create table film_with_previous (
  film_id serial primary key,
  title character varying(255) not null,

  previous integer references film_with_previous(film_id)
);
```

```
insert into film_with_previous (film_id, title, previous)
values
   (1, 'Terminator', null),
   (2, 'Terminator 2: Judgment Day', 1),
   (3, 'Terminator 3: Rise of the Machines', 2),
   (4, 'Terminator Salvation', 3),
   (5, 'Terminator Genisys', 1),
   (6, 'Terminator: Dark Fate', 2);
```

# ltree & more

```
with recursive saga as (
       select f.film_id, f.title, f.previous
       from film_with_previous f
       where f.film_id = 6
   union all
       select f.film_id, f.title, f.previous
       from film_with_previous f
       inner join saga s on s.previous = f.film_id
)
select title
from saga
order by film_id;
```

```
film_id |           title             | previous
--------+-----------------------------+----------
      1 | Terminator                  |
      2 | Terminator 2: Judgment Day  |        1
      6 | Terminator: Dark Fate       |        2
```


AUSTIN

# Foreign Data Wrapper

# Foreign Data Wrapper

```
create extension file_fdw;
create server file_server foreign data wrapper file_fdw;

create foreign table passwd (
   username text,
   pass text,
   uid int4,
   gid int4,
   gecos text,
   home text,
   shell text
) server file_server
options (format 'text', filename '/etc/passwd', delimiter ':', null '');
```

# Foreign Data Wrapper

```sql
select *
from passwd
where shell ~* '.*bash.*';
```

```
username | uid | gid |        home         |   shell
---------+-----+-----+---------------------+-----------
root     |   0 |   0 | /root               | /bin/bash
postgres | 999 | 999 | /var/lib/postgresql | /bin/bash
```

# Foreign Data Wrapper

```
create foreign table fs (
   inode text,
   block text,
   permission text,
   hard_link text,
   owner text,
   "group" text,
   size int,
   last_modification text,
   pathname text
) server file_server
options (format 'text', program '/list-all-files.sh', delimiter '|', null '');

select pathname, owner, "group", size
from fs
where permission ~ '.......rwx'
order by size desc;
```

| pathname | owner | group | size |
|---|---|---|---|
| /etc/alternatives/CREATE_TEXT_SEARCH_CONFIGURATION.7.gz | root | root | 71 |
| /etc/alternatives/ALTER_TEXT_SEARCH_CONFIGURATION.7.gz | root | root | 70 |
| /etc/alternatives/DROP_TEXT_SEARCH_CONFIGURATION.7.gz | root | root | 69 |
| /etc/alternatives/CREATE_TEXT_SEARCH_DICTIONARY.7.gz | root | root | 68 |
| /etc/alternatives/ALTER_TEXT_SEARCH_DICTIONARY.7.gz | root | root | 67 |

# Foreign Data Wrapper

# Foreign Data Wrapper

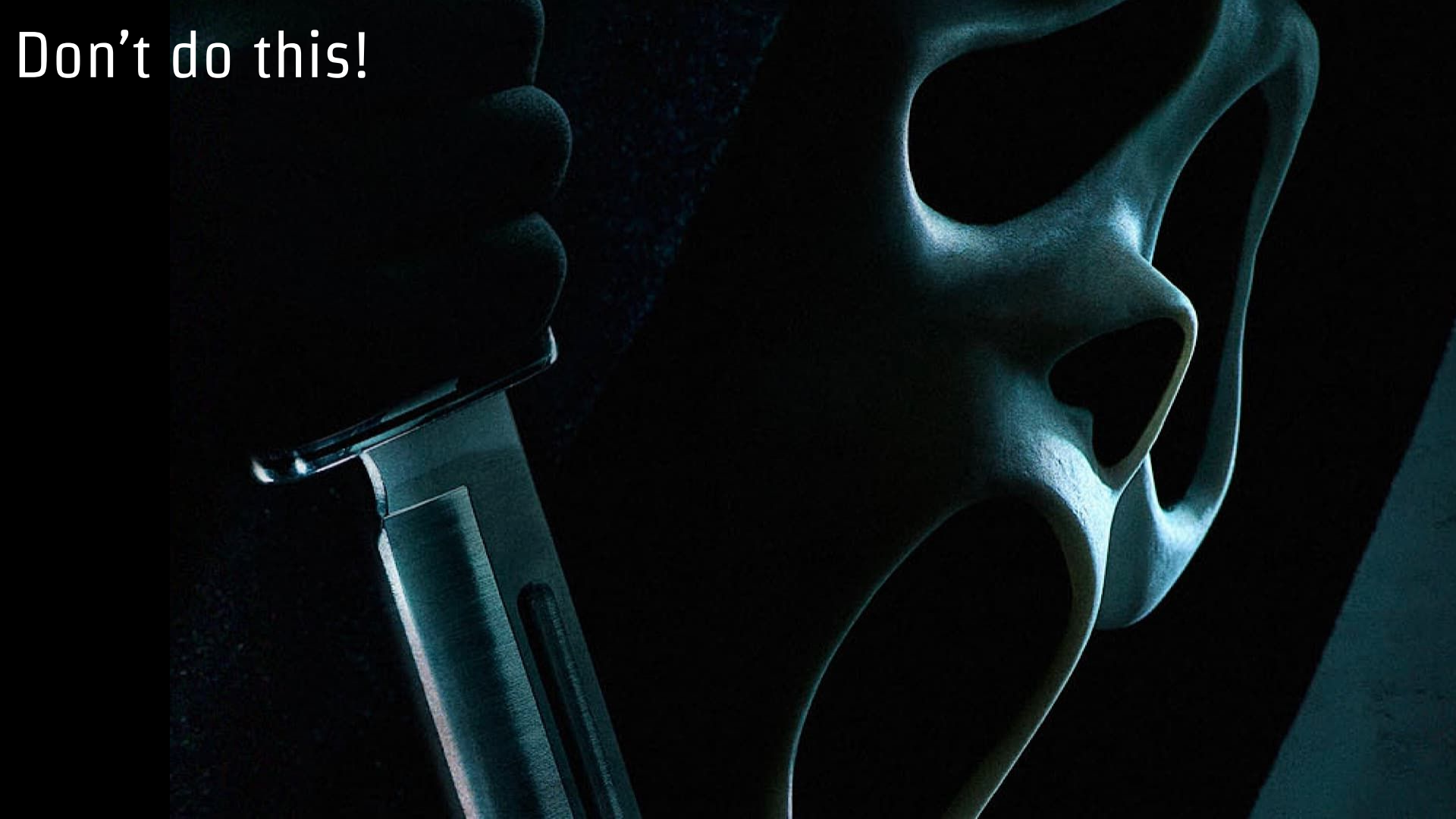
```
create extension postgres_fdw;

create server sakila
foreign data wrapper postgres_fdw
options (host 'amazing-pg', port '5432', dbname 'sakila');

create user mapping for postgres
server sakila
options (user 'postgres', password 'gjITg2bOO33DlbYju27wK2WoOlI2');
```

# Foreign Data Wrapper

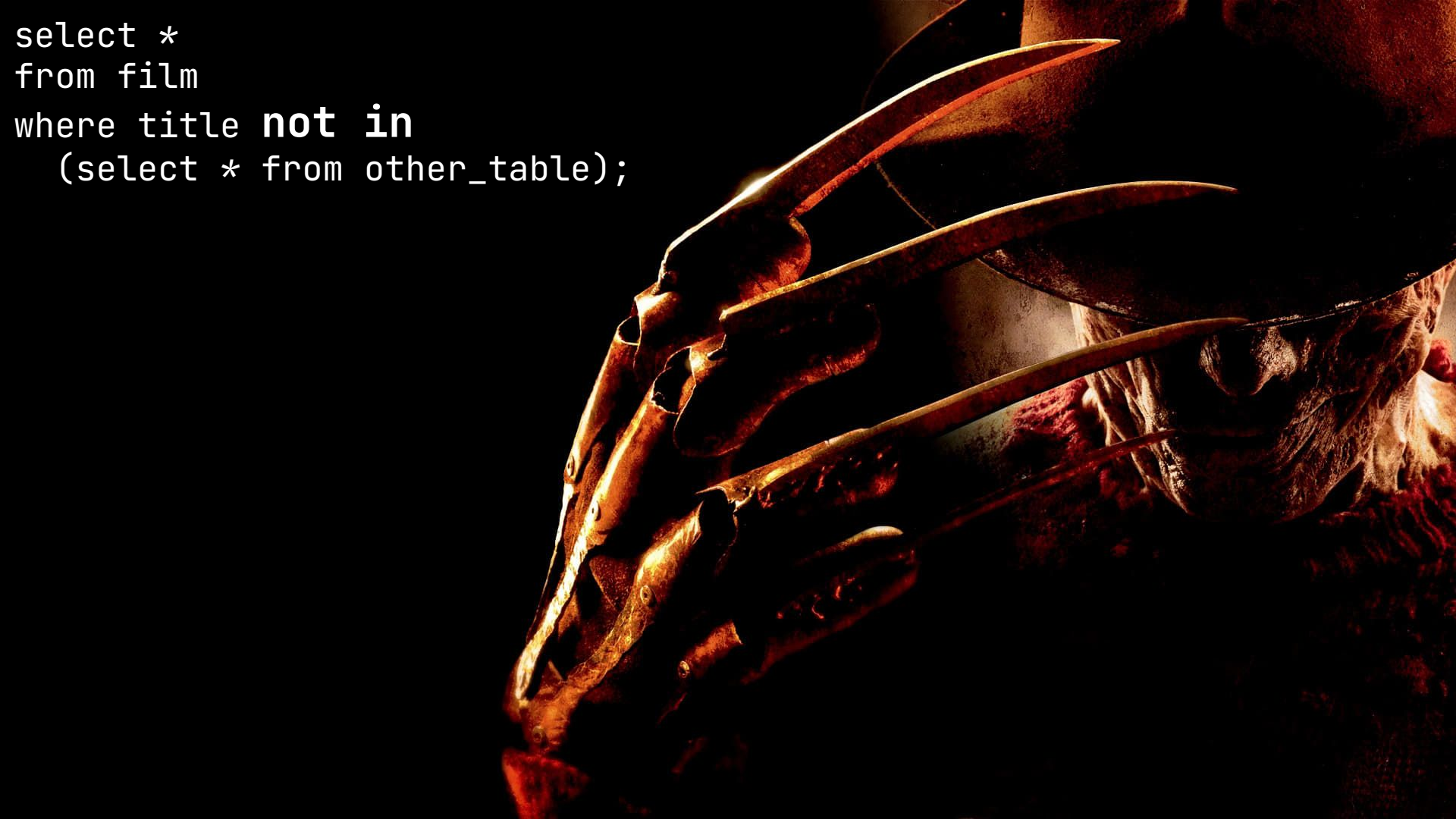
```
create foreign table remote_film (
   film_id integer not null,
   title character varying(255) not null
)
server sakila
options (schema_name 'public', table_name 'film');

select *
from remote_film;
```

```
   id    |      title
---------+------------------
       1 | ACADEMY DINOSAUR
       2 | ACE GOLDFINGER
       3 | ADAPTATION HOLES
       4 | AFFAIR PREJUDICE
       5 | AFRICAN EGG
```

# Don’t do this!



```
select *
from film
where title not in
  (select * from other_table);
```




create table xyz (
…
) inherits (film)

```
create table xyz (
 date timestamp
)
```

# But use them...

```
create table xyz (
   id uuid not null default uuid_generate_v4(),
   title text not null,
   fts_title tsvector generated always as (to_tsvector('english', title)) stored
)
```

```
-- Use PostGIS for geo-queries
create extension postgis;
```

```
create table xyz (
    json_data jsonb default '{}'
)
```

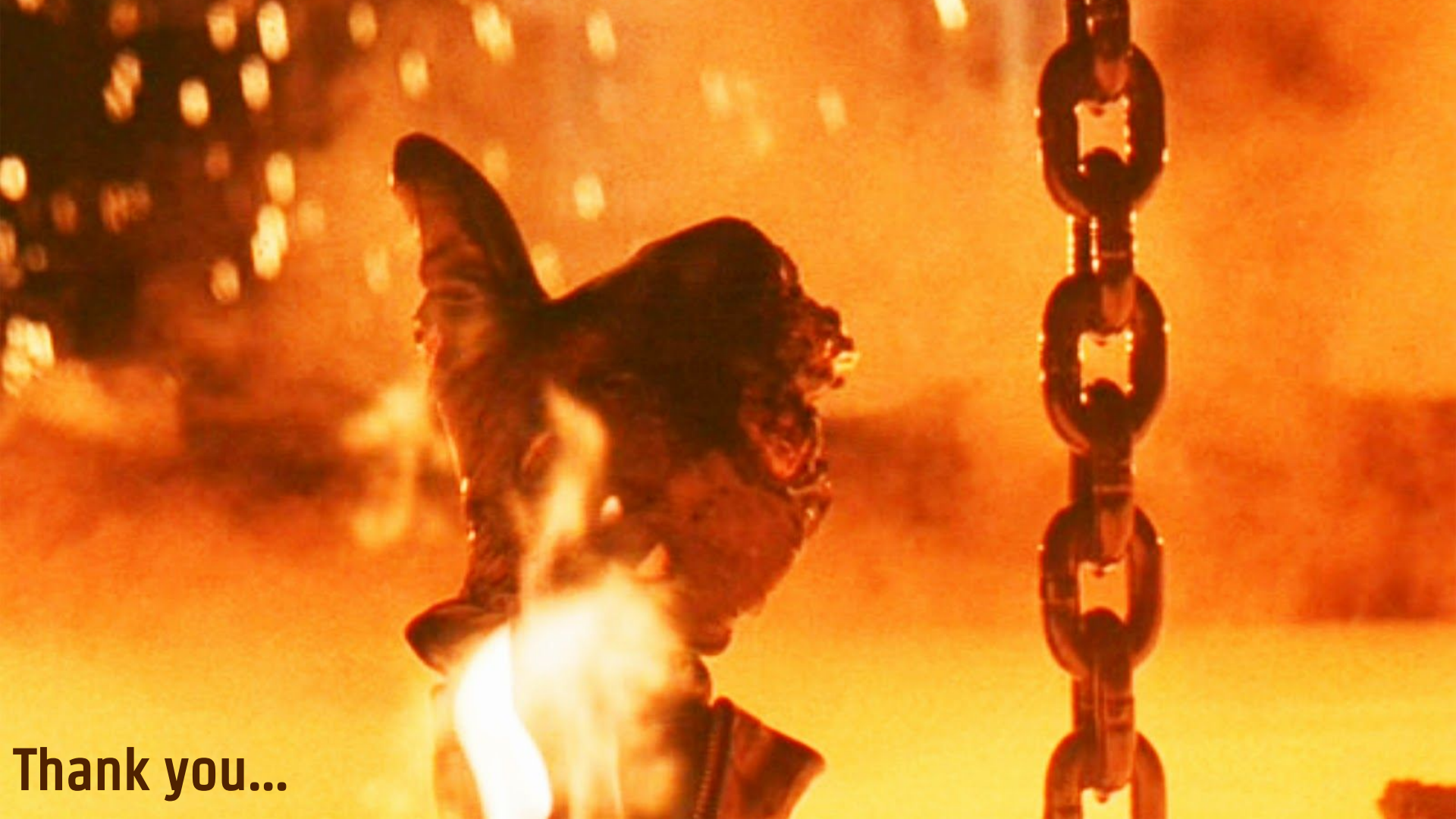

PostgREST
GRAPHJIN

# Everywhere!


In any server...

# In Kubernetes with...


Infinite possibilities...


Thank you...

# Questions?

@davinkevin@davinkevin.fr

# More SQL!

@davinkevin@davinkevin.fr

« JOOQ, joy of SQL » | Kevin Davin

11:10 - 12:00 | Conférence 3

Et si l'on reprenait le contrôle de nos interactions avec notre base de données préférée? Car en vrai, nos BDD sont des monstres de puissance qui sont souvent sous exploités 😓. Nous verrons, avec JOOQ comment écrire des requêtes SQL simples ou évolués, le tout facilement et de manière type-safe !

Venez découvrir cette superbe librairie OpenSource, compatible Java, Kotlin & Scala, qui va vous permettre de vous simplifier la vie et d'améliorer votre code... et même de découvrir de super fonctionnalités SQL que vous n'imaginiez même pas.

Languages | Conference (50 min) | Beginner | French