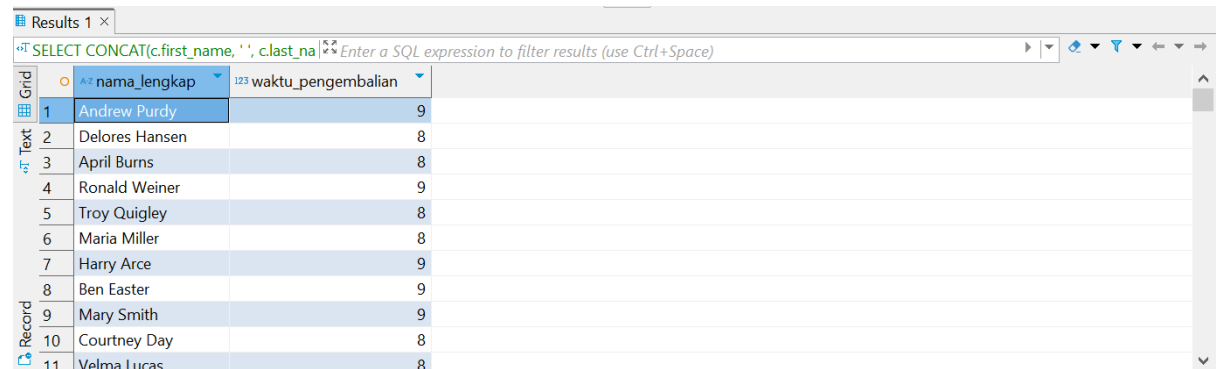


Pada skema dvdrental

1. Dapatkan nama lengkap customers dan waktu pengembalian untuk semua transaksi yang melebihi batas waktu pengembalian (7 hari).

```
SELECT CONCAT(c.first_name, ' ', c.last_name) AS nama_lengkap,  
EXTRACT(DAY FROM(r.return_date - r.rental_date)) AS waktu_pengembalian  
FROM customer c  
INNER JOIN rental r ON c.customer_id = r.customer_id  
WHERE r.return_date IS NOT NULL AND EXTRACT(DAY FROM(r.return_date -  
r.rental_date)) > 7;
```



Results 1 x

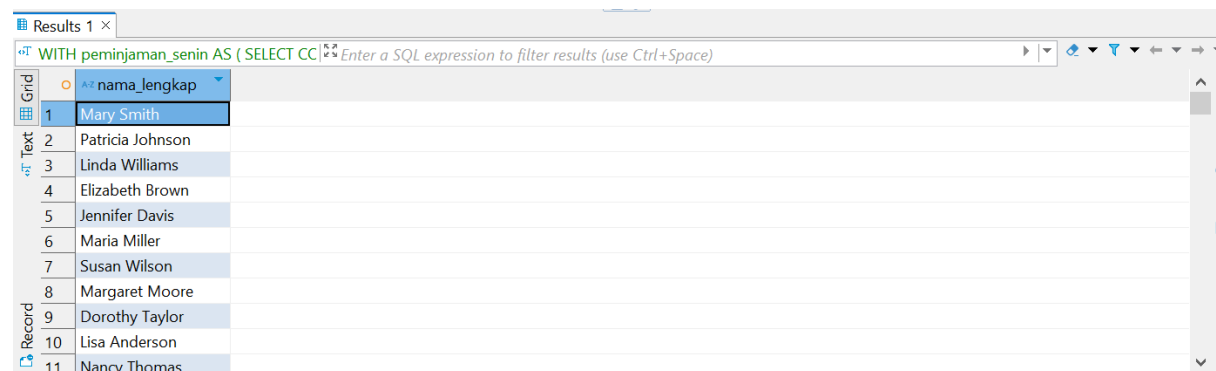
SELECT CONCAT(c.first\_name, ' ', c.last\_name) AS nama\_lengkap, EXTRACT(DAY FROM(r.return\_date - r.rental\_date)) AS waktu\_pengembalian

Grid	nama_lengkap	waktu_pengembalian
1	Andrew Purdy	9
2	Delores Hansen	8
3	April Burns	8
4	Ronald Weiner	9
5	Troy Quigley	8
6	Maria Miller	8
7	Harry Arce	9
8	Ben Easter	9
9	Mary Smith	9
10	Courtney Day	8
11	Velma Lucas	8

2. Tampilkan nama pelanggan yang melakukan transaksi peminjaman lebih dari sekali pada hari Senin! Gunakan CTE!

– Menggunakan CTE

```
WITH peminjaman_senin AS (  
    SELECT  
        CONCAT(c.first_name, ' ', c.last_name) AS nama_lengkap,  
        COUNT(r.rental_id) as jumlah_peminjaman  
    FROM  
        customer c  
    INNER JOIN  
        rental r ON c.customer_id = r.customer_id  
    WHERE  
        EXTRACT(DOW FROM r.rental_date) = 1 -- 1 adalah hari senin  
    GROUP BY  
        c.customer_id, c.first_name, c.last_name  
)  
SELECT  
    nama_lengkap  
FROM  
    peminjaman_senin  
WHERE jumlah_peminjaman > 1;
```



Results 1 x

WITH peminjaman\_senin AS ( SELECT CC, ... )

Grid	nama_lengkap
1	Mary Smith
2	Patricia Johnson
3	Linda Williams
4	Elizabeth Brown
5	Jennifer Davis
6	Maria Miller
7	Susan Wilson
8	Margaret Moore
9	Dorothy Taylor
10	Lisa Anderson
11	Nancy Thomas

3. Temukan nama aktor dan jumlah film yang dimainkan, serta peringkat aktor berdasarkan jumlah film. Urutkan berdasarkan peringkat secara ascending. Gunakan RANK!

```
SELECT CONCAT(a.first_name, ' ', a.last_name) AS nama_lengkap,
COUNT(fa.film_id) as jumlah_film, RANK() OVER (ORDER BY count(fa.film_id) ASC)
AS ranked
FROM actor a
LEFT JOIN film_actor fa on a.actor_id = fa.actor_id
GROUP BY a.actor_id
ORDER BY ranked ASC;
-- supaya semua aktor terhitung maka memakai LEFT JOIN (aktor yang belum
pernah tampil dalam film apa pun akan tetap terdaftar, dengan jumlah film 0)
```

Results 1 x Statistics 1			
Enter a SQL expression to filter results (use Ctrl+Space)			
	nama_lengkap	jumlah_film	ranked
1	Emily Dee	14	1
2	Julia Fawcett	15	2
3	Judy Dean	15	2
4	Julia Zellweger	16	4
5	Adam Grant	18	5
6	Sissy Sobieski	18	5
7	Russell Close	19	7
8	Sandra Peck	19	7
9	Penelope Guinness	19	7
10	Cameron Wray	19	7
11	Christopher Penn	20	11

Pada skema DS Salaries

4. Tampilkan (semua kolom) dengan job\_title yang memiliki salary\_in\_usd lebih besar dari rata-rata salary dari seluruh job\_title. Namun, tampilkan hanya company\_size = S.

```
SELECT *
FROM ds_salaries
WHERE
    salary_in_usd > (select avg(salary_in_usd) from ds_salaries)
    AND company_size = 'S';
-- Memakai AVG(salary_in_usd) karena supaya satuan mata uangnya sama
-- Jadi kriterianya salary_in_usd yang lebih besar dari rata-rata salary
dengan mata uang usd
```

ds_salaries 1 x Statistics 1							
Enter a SQL expression to filter results (use Ctrl+Space)							
	id	work_year	experience_level	employment_type	job_title	salary	salary_in_usd
1	1	2,020	SE	FT	Machine Learning Scientist	260,000	260,000
2	6	2,020	SE	FT	Lead Data Scientist	190,000	190,000
3	9	2,020	SE	FT	Lead Data Engineer	125,000	125,000
4	17	2,020	SE	FT	Big Data Engineer	100,000	114,040
5	39	2,020	EN	FT	Machine Learning Engineer	138,000	138,000
6	107	2,021	SE	FT	Data Engineer	115,000	115,000
7	126	2,021	SE	FT	Machine Learning Scientist	120,000	120,000
8	149	2,021	SE	FT	Cloud Data Engineer	160,000	160,000
9	150	2,021	SE	FT	Director of Data Science	168,000	168,000
10	159	2,021	EN	FT	Machine Learning Engineer	125,000	125,000