

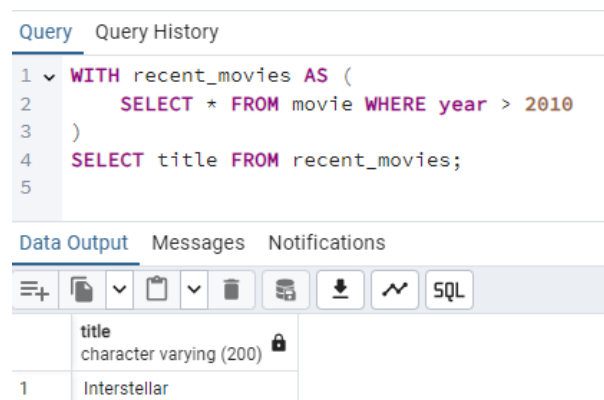
Part 3

CTE (Common Table Expression)

CTE adalah pernyataan sementara yang mendefinisikan hasil query seolah-olah seperti sebuah tabel sementara yang bisa digunakan dalam query utama.

Fungsi Utama CTE:

- Memecah query kompleks menjadi bagian yang lebih mudah dibaca.
- Dapat dipakai untuk rekursi, agregasi, atau multi-step query.
- Membantu meningkatkan readability dan modularitas query.



The screenshot shows a SQL IDE interface. The top tab is 'Query', and the 'Query History' pane displays a CTE query:

```
1 WITH recent_movies AS (  
2   SELECT * FROM movie WHERE year > 2010  
3 )  
4 SELECT title FROM recent_movies;  
5
```

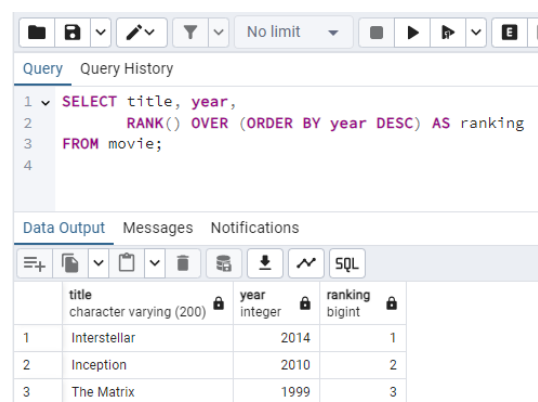
Below the query editor, the 'Data Output' tab is active, showing the results of the query. The output is a table with one column, 'title', and one row, 'Interstellar'.

	title character varying (200)
1	Interstellar

Window Function

Window Function adalah fungsi yang melakukan perhitungan di atas sekumpulan baris terkait (window) tanpa mengelompokkan data seperti pada GROUP BY.

Artinya, setiap baris tetap ditampilkan, tetapi ada nilai tambahan yang dihitung berdasarkan baris terkait.



The screenshot shows a SQL IDE interface. The top tab is 'Query', and the 'Query History' pane displays a window function query:

```
1 SELECT title, year,  
2   RANK() OVER (ORDER BY year DESC) AS ranking  
3 FROM movie;  
4
```

Below the query editor, the 'Data Output' tab is active, showing the results of the query. The output is a table with three columns: 'title', 'year', and 'ranking'.

	title character varying (200)	year integer	ranking bigint
1	Interstellar	2014	1
2	Inception	2010	2
3	The Matrix	1999	3

Semua film ditampilkan, dengan kolom tambahan ranking berdasarkan tahun rilis (dari terbaru ke terlama).

Create VIEW and make them into table with indexing

Create view

VIEW adalah virtual table yang menyimpan query sebagai objek database. View tidak menyimpan data secara fisik, melainkan menampilkan hasil dari query saat view diakses.

The screenshot shows a SQL IDE interface with a toolbar at the top. The 'Query' tab is active, displaying a SQL query to create a view named 'movie_actor_view'. The query is as follows:

```
1 CREATE VIEW movie_actor_view AS
2 SELECT m.title, m.year, a.name AS actor_name
3 FROM movie m
4 JOIN movie_actor ma ON m.id = ma.movie_id
5 JOIN actor a ON ma.actor_id = a.id;
6
```

Below the query editor, the 'Data Output' tab is selected, showing the message: 'CREATE VIEW' and 'Query returned successfully in 171 msec.'

The screenshot shows the same SQL IDE interface, but now the 'Data Output' tab displays the results of a query executed against the 'movie_actor_view'. The query is 'SELECT * FROM movie_actor_view;'. The results are shown in a table with three columns: 'title', 'year', and 'actor_name'. The data is as follows:

	title character varying (200)	year integer	actor_name character varying (100)
1	Interstellar	2014	Matthew McConaughey
2	Inception	2010	Leonardo DiCaprio
3	The Matrix	1999	Keanu Reeves

Mengubah View menjadi Tabel

The screenshot shows a SQL IDE interface with a toolbar at the top. The 'Query' tab is active, displaying a SQL query to create a table named 'movie_actor_table' from the 'movie_actor_view'. The query is as follows:

```
1 CREATE TABLE movie_actor_table AS
2 SELECT * FROM movie_actor_view;
3
```

Below the query editor, the 'Data Output' tab is selected, showing the message: 'SELECT 3' and 'Query returned successfully in 137 msec.'

Menambahkan Index

The screenshot shows a SQL IDE interface with a toolbar at the top. The 'Query' tab is active, displaying a SQL query to create an index named 'idx_actor_name' on the 'movie_actor_table' table, specifically on the 'actor_name' column. The query is as follows:

```
1 CREATE INDEX idx_actor_name ON movie_actor_table (actor_name);
2
```

Below the query editor, the 'Data Output' tab is selected, showing the message: 'CREATE INDEX' and 'Query returned successfully in 164 msec.'