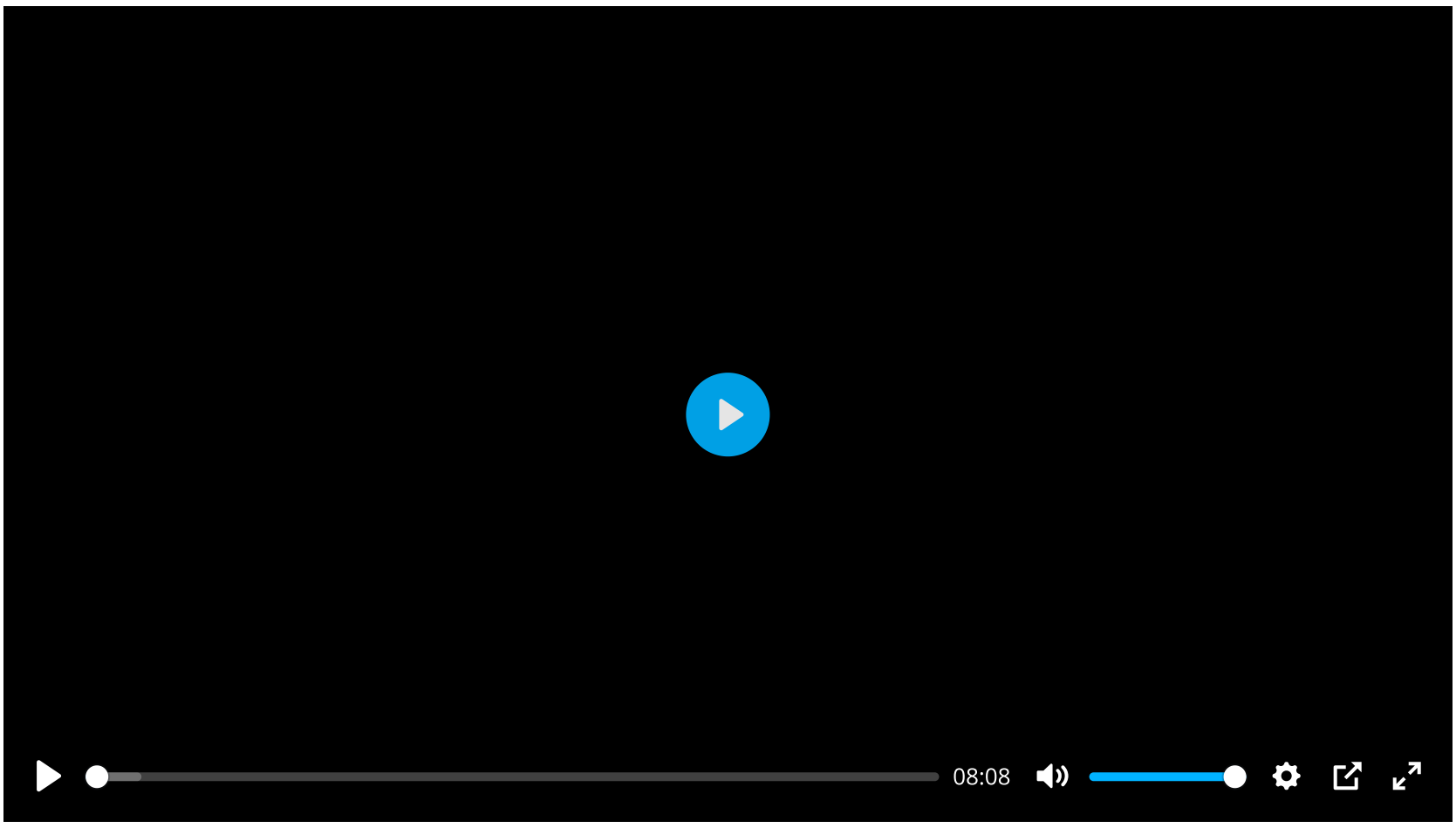
5.4 База данных глазами Data Engineer'a



Текстовая расшифровка видео:

ОБЩИЕ ТАБЛИЧНЫЕ ВЫРАЖЕНИЯ

План:

- Common Table Expressions;
- Кейс;
- Найти суммарные и средние продажи;
- Сольем все воедино: subquery;
- Сольем все воедино: CTE.

Common Table Expressions

Common Table Expressions – это общие табличные выражения.

Основная идея: мы декларативно описываем, что хотим получить. БД в зависимости от ее внутренней реализации решает, каким образом нам это вернуть. Мы не расписываем, как именно хотим получить эти данные, а пишем, что хотим получить.

С одной стороны подобное очень удобно, с другой – многим не хватает переменной.

Common Table Expressions использует ключевое слово «WITH». Это чем-то схоже с «WITH» в Python. Как вы помните, «WITH» в Python создает контекст. Например, мы открываем файл, создается переменная, когда же мы выходим, переменная автоматически уничтожается, контекст закрывается, файл закрывается.

В SQL похожая ситуация: с помощью «WITH» мы создаем контекст, создавая из запросов переменные relation'ы, которые будут жить, пока исполняется контекст. Мы присваиваем их результаты переменным. Внутри тела Common Table Expressions мы можем обращаться к переменным несколько раз, писать Select'ы.

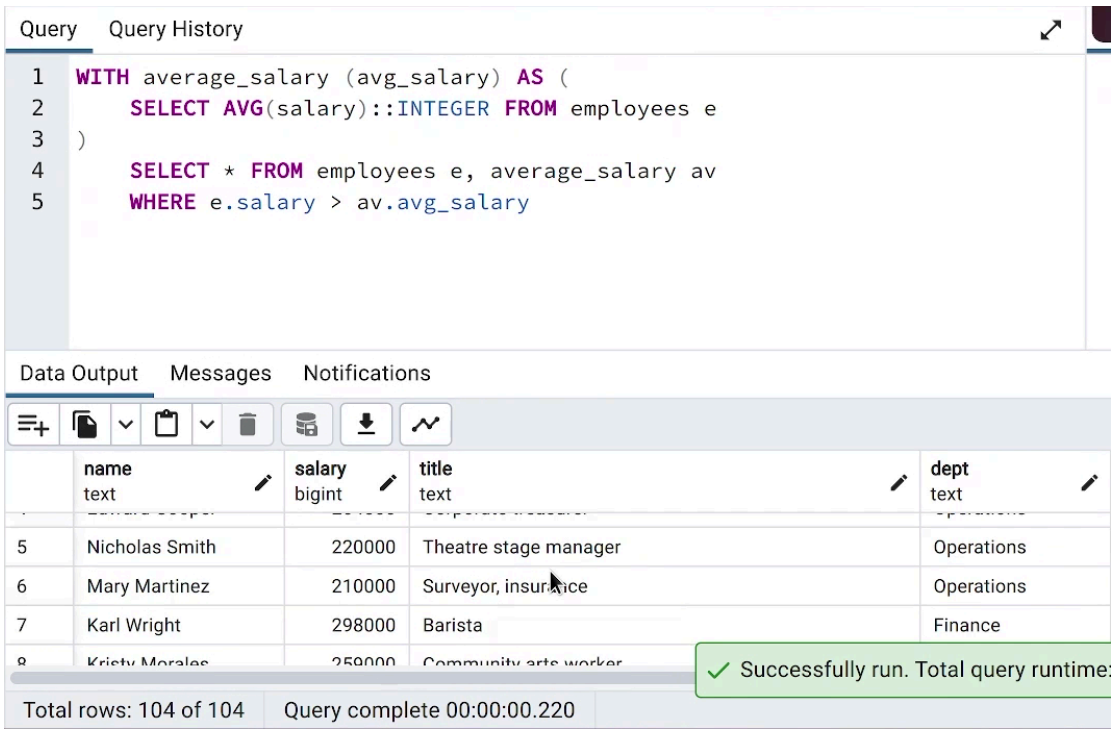
- Не персистятся на диск, живут только в рамках запроса;

- По логике очень похожи на подзапросы (subquery);
- Могут переиспользоваться много раз внутри блока.

Рассмотрим пример:

```
WITH average_salary (avg_salary) AS (
    SELECT AVG(salary)::INTEGER FROM employees e
)
    SELECT * FROM employees e, average_salary av
    WHERE e.salary > av.avg_salary
```

Это стандартный запрос с агрегацией. Мы считаем среднюю зарплату и выбираем всех сотрудников, у которых зарплата выше средней. Этот запрос можно достаточно легко переписать без WITH, он будет работать:



Кейс

Рассмотрим более репрезентативный пример. Представим, что мы хотим решить подобную задачу:

Есть данные о продажах в магазинах. Необходимо найти магазины, в которых суммарные продажи были выше, чем средние между всеми магазинами.

По шагам:

- Найти суммарные продажи по каждому магазину (total_sales);
- Найти средние продажи по всем магазинам (average_sales);
- Найти магазины, в которых суммарные продажи больше средних.

Найти суммарные и средние продажи

Если буквально следовать шагам, то получим следующее:

Сначала мы считаем продажи и суммируем по магазину, обсчитывая стандартным group by:

```
SELECT s.store_id, SUM(cost) AS total_sales_per_store
FROM sales s GROUP BY s.store_id
```

Если мы хотим посчитать среднее, мы можем переиспользовать этот запрос в виде subquery:

```
SELECT avg(total_sales_per_store) AS avg_sales_for_all_stores FROM (
    SELECT s.store_id, SUM(cost) AS total_sales_per_store
    FROM sales s GROUP BY s.store_id
) x
```

Сольем все воедино: subquery

Теперь нужно выбрать из первого и второго, где общие продажи по магазину выше средних, которые мы посчитали:

```
SELECT * FROM
    (SELECT s.store_id, SUM(cost) AS total_sales_per_store
     FROM sales s GROUP BY s.store_id) total_sales
JOIN (SELECT avg(total_sales_per_store) AS avg_sales_for_all_stores
      FROM (SELECT s.store_id, SUM(cost) AS total_sales_per_store
            FROM sales s GROUP BY s.store_id) x
    ) avg_sales
ON total_sales.total_sales_per_store > avg_sales.avg_sales_for_all_stores
```

Здесь получаются «танцы с бубном», так как делаем select. Вы можете заметить хитрость: мы сделали через join со сравнением (мы берем не по равенству полей, а по полю).

Если приглядеться, можно увидеть, что мы делаем один и тот же запрос несколько раз. Появляется желание переписать более красиво.

Сольем все воедино: CTE

Если мы перепишем подобное с использованием CTE, то получим лучший вариант:

```
WITH total_sales (store_id, total_sales_per_store) AS
    (SELECT s.store_id, SUM(cost) AS total_sales_per_store
    FROM sales s GROUP BY s.store_id),
    avg_sales (avg_sales_for_all_stores) AS
    (SELECT avg(total_sales_per_store) AS avg_sales_for_all_stores
    FROM total_sales)
SELECT * FROM total_sales ts JOIN avg_sales avs
ON ts.total_sales_per_store > avs.avg_sales_for_all_stores
```

Сначала мы считаем среднее и присваиваем это переменной. Далее, в рамках «WITH» можем ссылаться на предыдущий подсчитанный результат. Мы считаем средние зарплаты по тому, что только что посчитали, переиспользуя.

У нас получаются две переменные, а также avg sales и total sales. В рамках данного «WITH» нам достаточно сделать базовый join и получить тот же результат.

Query будет работать так же, как предыдущий, при этом будет более читаемым и понятным.