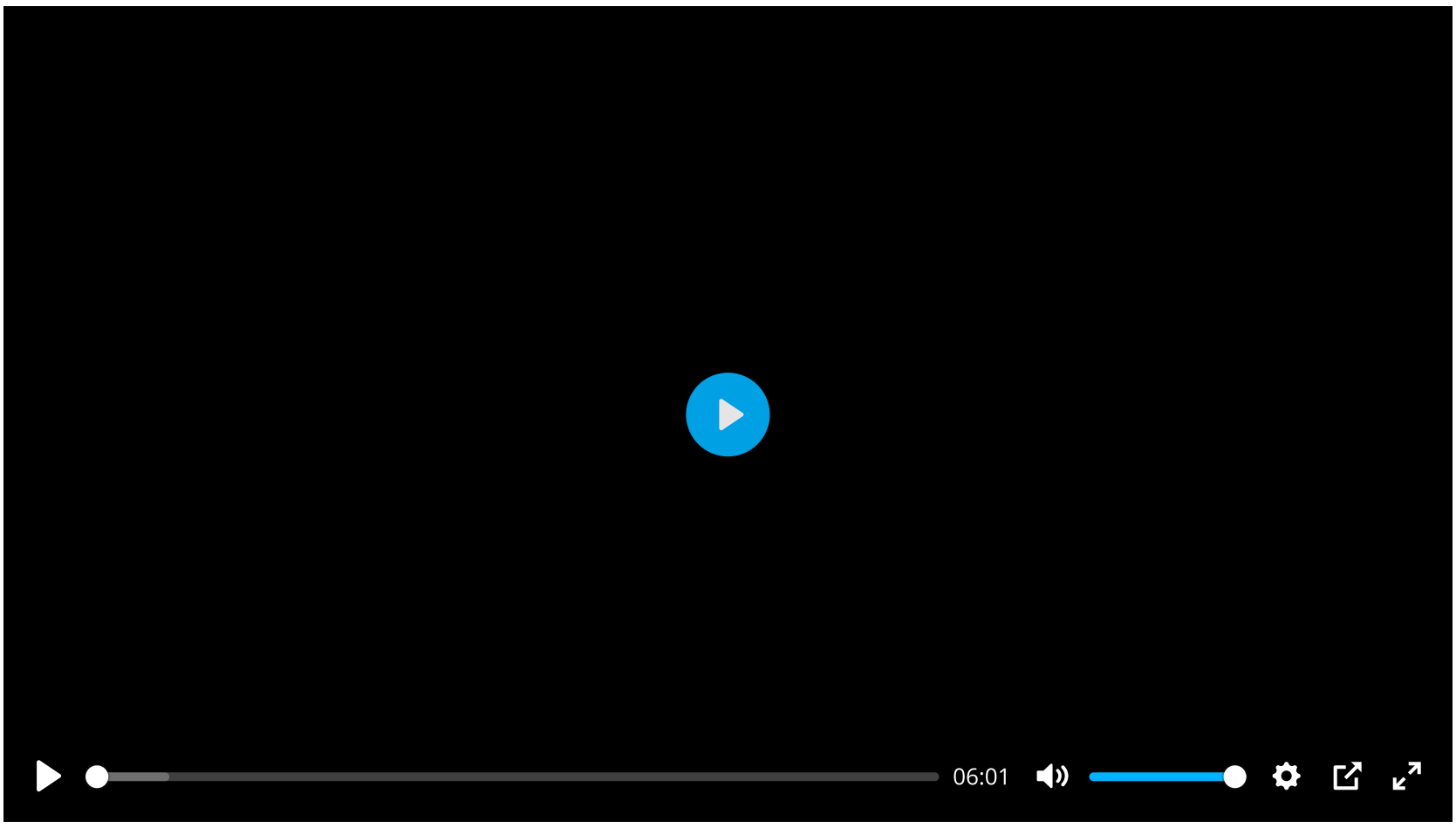
5.3 SQL. Принципы ACID.



Текстовая расшифровка видео:

АГРЕГАЦИЯ И ПОДЗАПРОСЫ

План:

- Считаем выгоду;
- Yo Dawg, мы положили запрос в твой запрос.

Считаем выгоду

Вспомним о существовании такого понятия, как «**Having**».

Когда мы делаем группировку, мы можем использовать «When», но он фильтрует только первую выборку. Если же мы хотим проводить фильтрацию в группах, нужно использовать «Having».

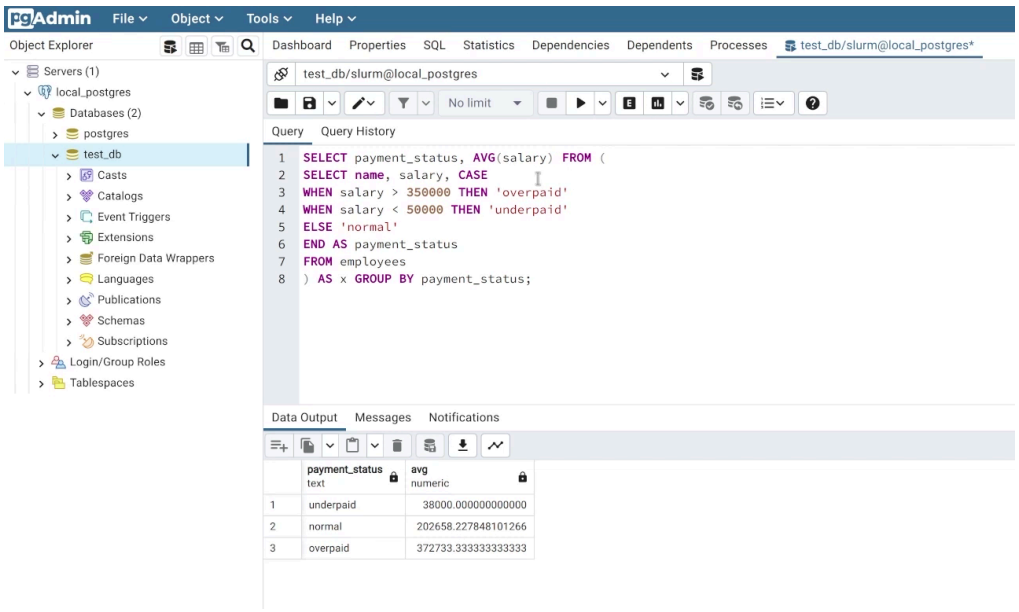
Почитать про HAVING

Почему не просто WHERE

Группировка и агрегация:

```
SELECT fired, SUM(salary) FROM employees GROUP BY fired HAVING fired = TRUE;
```

Мы можем использовать подзапрос. Его мы можем вставить в любое место нашего query. Его можно использовать и в «Where», и в «Having», и с ограничениями внутри агрегата, также можно использовать во «From» (мы генерируем временную табличку и делаем выборку напрямую из нее):




```
SELECT payment_status, AVG(salary) FROM (
            SELECT name, salary, CASE
            WHEN salary > 350000 THEN 'overpaid'
            WHEN salary < 50000 THEN 'underpaid'
            ELSE 'normal'
            END AS payment_status
            FROM employees
) AS x GROUP BY payment_status;
```

Обратите внимание, что мы группируем здесь по колонке, которую сами вычислили.

Yo Dawg, мы положили запрос в твой запрос

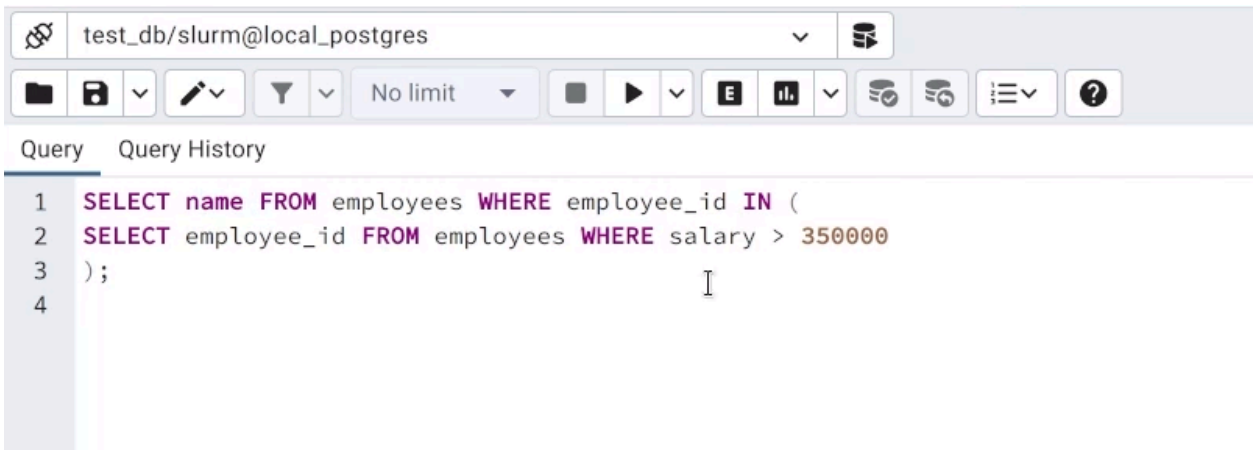
«Where» – другой вариант использования подзапроса.

Рассмотрим задачу:

```
SELECT name FROM employees WHERE employee_id IN (
            SELECT employee_id FROM employees WHERE salary > 350000
);
```

Мы достаем всех сотрудников, у которых зарплата больше 350000. При этом достать хотим только айдишник. Мы могли бы написать через «Where», но иногда в некоторых ситуациях удобнее оперировать айдишниками (в том числе при оптимизации запроса).

Выполнив данный запрос, получаем следующее:



Получаем список имен всех сотрудников, у которых зарплата достаточно высокая. Сначала мы создали виртуальную табличку и уже из нее выбрали.

Отсюда вытекает важное правило о том, как работает SQL и реляционная алгебра: **результат любого запроса – это relation.**

Можно составлять запросы большой степени множности.

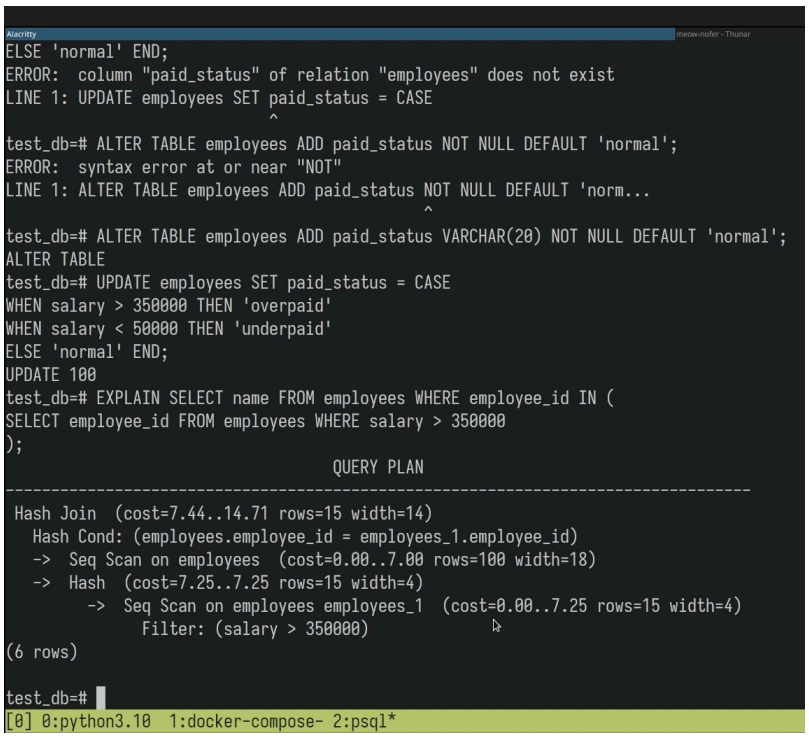
Плюс:

- Некоторые задачи гораздо проще решить с использованием подзапросов.

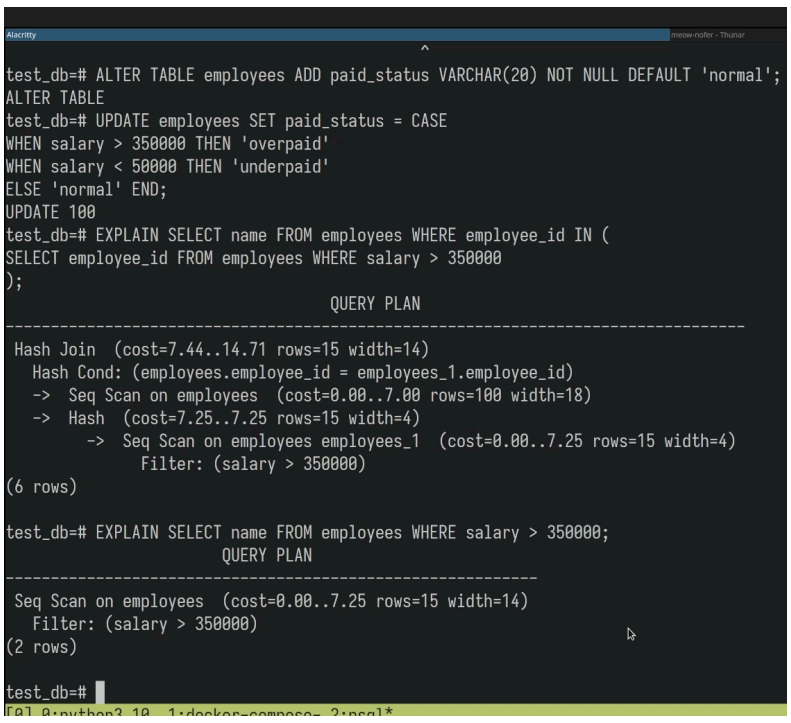
Минус:

- Злоупотребление подзапросами может сильно затруднять работу оптимизатора.

Рассмотрим пример:



Работа проходит неэффективно: происходит join таблички саму на себя. Напишем запрос напрямую:



Это будет линейный скан без дополнительных сложностей.

Как вам урок?

Изучил, далее ›