```sql
/* Query projetada que seleciona o maior salario dentre todos os funcionarios que foram admitidos em um perído específico */
explain SELECT MAX(salary) AS salary
FROM employee
WHERE date_admission > '01-01-2019'
```

QUERY PROJETADA SEM SUBQUERY

Plano de Consulta:

	QUERY PLAN text
1	Aggregate (cost=60.40..60.41 rows=1 width=32)
2	-> Append (cost=0.00..59.27 rows=453 width=32)
3	-> Seq Scan on employee (cost=0.00..0.00 rows=1 width=32)
4	Filter: (date_admission > '2019-01-01'::date)
5	-> Seq Scan on director (cost=0.00..14.25 rows=113 width=32)
6	Filter: (date_admission > '2019-01-01'::date)
7	-> Seq Scan on manager (cost=0.00..14.25 rows=113 width=32)
8	Filter: (date_admission > '2019-01-01'::date)
9	-> Seq Scan on especialist (cost=0.00..14.25 rows=113 width=32)
10	Filter: (date_admission > '2019-01-01'::date)
11	-> Seq Scan on analyst (cost=0.00..14.25 rows=113 width=32)
12	Filter: (date_admission > '2019-01-01'::date)

A consulta projetada sem subquery, ao contrário da projetada que contém subquery, não demora tanto quanto para ser executada. Aqui também é feito o uso do “filter” com a data de admissão e a busca sequencial (“seq scan”), porém não mais em paralelo (“parallel seq scan”) por conta de não ter outra “rotina” em execução. Apesar dessa consulta ter um melhor resultado em termos de custo ainda é possível afirmar que caso nosso modelo estivesse com mais dados, ou por exemplo, fosse um modelo focado para o big data, este cenário poderia até não chegar a retornar algo efetivo por sempre ser necessário realizar um “filtro” para depois realizar uma busca sequencial.