

Structured Query Language

# SQL





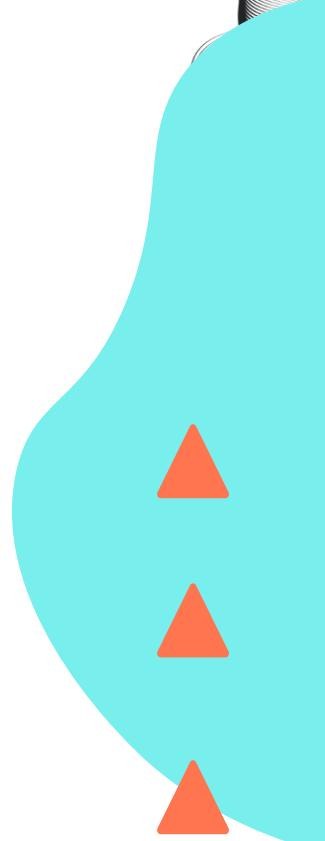
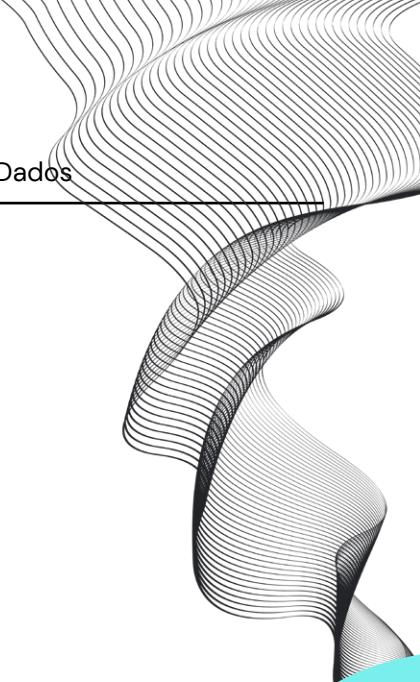
# Jornada do curso

01 Básico

02 Intermediário

03 Indo além

04 Processo de ETL com SQL





## Agenda

- 01 ► Quando é preciso criar separações:  
**CASE WHEN**
- 02 ► O WHERE não funciona: HAVING
- 03 ► Otimização de consultas:  
**SUBQUERY & TEMPORARY TABLE**
- 04 ► Analisar datas: DATE & SUBSTR



**70%** dos negócios no mundo estão coletando dados mais rapidamente do que conseguem utilizá-los.

Apenas **21%** das empresas sabem tratar os dados adequadamente.

*Fonte: Data Paradox, Dell Technologies, 2020.*



## Missão da aula

Aprofundar o conhecimento de SQL a partir de funções que permitam uma análise mais detalhada e otimizada dos dados.



— Isto é impossível.  
— Só se você acreditar que é. Às vezes,  
eu acredito em seis coisas impossíveis  
antes do café da manhã.”

Lewis Carroll, autor de Alice no País das Maravilhas.



# Mão na massa

A Conquer Sales recebeu uma reclamação de que, no catálogo, existem poucas categorias que vendem produtos da cor cinza. Então, seu chefe pede para você levantar quais os produtos que têm a cor cinza na descrição e marcar também a categoria.

- ▶ Em quantas categorias diferentes há produtos cinza?
- ▶ Se agrupar as categorias “Furniture” (móveis), “Office Supplies” (materiais de escritório) e “Outros”, quantos produtos com a cor cinza na descrição terá no total?
- ▶ Ainda sobre o catálogo da Conquer Sales, quantos produtos diferentes tem por categoria na cor cinza?

---

## Agenda

- O1** ▶ Quando é preciso criar separações:  
CASE WHEN
- O2** ▶ O WHERE não funciona: HAVING
- O3** ▶ Otimização de consultas:  
SUBQUERY & TEMPORARY TABLE
- O4** ▶ Analisar datas: DATE & SUBSTR



## WHERE vs HAVING

O comando **WHERE** foi estudado no módulo 1 do curso – ele determina que o banco de dados pesquise e selecione apenas os dados que se encaixem em uma limitação ou condição. No entanto, esse comando não funciona em todas as consultas em SQL, apenas naquelas que utilizam as colunas originais da planilha.

Caso você queira construir uma limitação com base em uma coluna que você criou usando alguma função de agregação (COUNT, SUM etc.) ou agrupando por meio do **GROUP BY**, é necessário usar a função **HAVING**, como na estrutura a seguir:

```
SELECT COUNT(coluna1), coluna2
FROM tabela1
GROUP BY coluna2
HAVING COUNT(coluna1) > 5;
```

## Mão na massa

O marketing da empresa recebeu sua base, fez a campanha, mas não teve o resultado esperado. Por isso, a equipe decidiu fazer uma campanha mais abrangente, incluindo os estados em que tiveram no mínimo 30 e no máximo 100 vendas.

**Em quais estados houve entre 30 e 100 vendas?**



# Pulo do gato

Use **WHERE** se você quiser primeiro filtrar e depois agrupar.

Use **HAVING** se você quiser primeiro agrupar e depois filtrar.

Para quase todos os SQLs que citamos, o **HAVING** virá depois do **GROUP BY**. Então, fica a dica: o **WHERE** filtra o que você pede antes de agregar, e o **HAVING** filtra o que você pede depois de fazer o agrupamento dos dados.

---

## Agenda

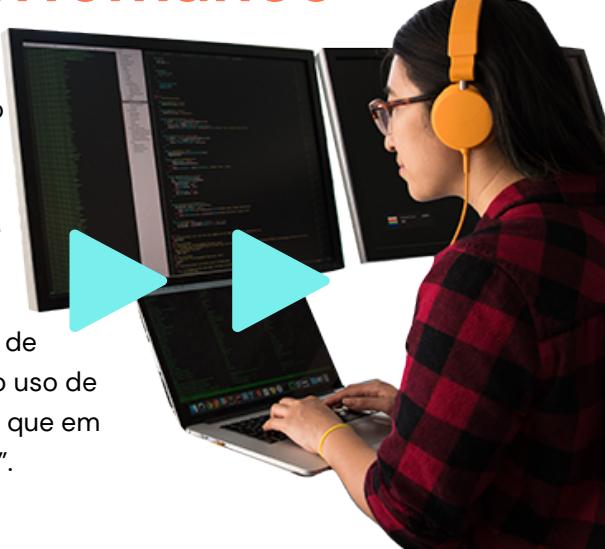
- 01 ► Quando é preciso criar separações:  
CASE WHEN
- 02 ► O WHERE não funciona: HAVING
- 03 ► Otimização de consultas:  
**SUBQUERY & TEMPORARY TABLE**
- 04 ► Analisar datas: DATE & SUBSTR



# dado pré-filtrado = ganho em performance

Quando o tamanho é importante, começar a fazer o enriquecimento de informações a partir do seu dado recortado, ou “pré-filtrado”, traz muito ganho em performance para a sua consulta.

Podemos fazer essa pré-filtragem de duas maneiras em SQL: através do uso de **SUBQUERY** e **TEMPORARY TABLE**, que em inglês significa “tabela temporária”.



## SUBQUERY

É a consulta dentro da consulta, ou um **SELECT** dentro de outro **SELECT**. Também conhecida como *subselect* ou *subconsulta* – justamente por criar uma consulta ou um *select* dentro de outro comando. Esse tipo de consulta pode ser estruturada da seguinte maneira:

```
SELECT *
FROM tabela1 AS T
WHERE coluna1 IN (SELECT coluna2 FROM tabela2)
```



## TEMPORARY TABLE

**TEMPORARY TABLE** é uma tabela temporária e, como o nome sugere, não existe de forma permanente na sua base. Ela é criada quando o usuário precisa daquela informação apenas por um período específico de tempo, como uma sessão.

Existem três tipos de tabelas temporárias:

- > **locais**, acessadas apenas por quem as criou naquela sessão;
- > **globais**, acessadas por qualquer sessão;
- > **variáveis**, que se assemelham às locais por serem visíveis apenas na sessão em que são criadas; no entanto, apresentam outras limitações – o uso do comando **DECLARE** e a invisibilidade nos *batches* subsequentes à criação da tabela.

# Nada é tão bom que não possa melhorar.

Você pode até montar consultas sem usar um desses recursos (**SUBQUERY** ou **TEMPORARY TABLE**), mas é inegável que, ao usá-los, tudo se torna **mais organizado** e, por consequência, **mais legível**. Pensando ainda em bases maiores, sua query trará resultados em um tempo muito menor, elevando a **eficiência** da sua consulta.





# Mão na massa

Usando como base os últimos levantamentos que você apresentou, seu chefe pediu um levantamento do estado em que há mais clientes na categoria de produtos mais caros.

- ▶ Quais são as 3 cidades com o maior valor de venda em tecnologia no estado da Califórnia?
  - ▶ Quais são os 10 produtos mais vendidos no estado da Califórnia?
- 

# Pulo do gato

Teste o tempo de execução com um **LIMIT** para escolher entre **SUBQUERY** e **TEMPORARY TABLE**.

Se, em seu local de trabalho, você tiver acesso a ambos, **SUBQUERY** e **TEMPORARY TABLE**, vale testar o tempo de execução com um **LIMIT** de 1.000 linhas. Assim, se o que você estiver buscando for maior do que isso, você terá esse teste para entender qual a solução mais eficiente em tempo para os dados com os quais você está trabalhando.



## Agenda

- 01 ► Quando é preciso criar separações:  
CASE WHEN
- 02 ► O WHERE não funciona: HAVING
- 03 ► Otimização de consultas:  
SUBQUERY & TEMPORARY TABLE
- 04 ► Analisar datas: DATE & SUBSTR

## Como analisar um determinado período no tempo?

---

A linguagem SQL nos permite consultar o banco de dados com um viés temporal, isto é, realizar uma determinada consulta considerando um intervalo de tempo. Isso é extremamente útil, pensando que empresas, no geral, querem analisar resultados específicos (de um dia, mês, quarter, semestre ou ano).

Para realizar esse tipo de análise, podemos utilizar diferentes funções de dados, que podem

nos fornecer vários insights de negócio:

- › Em que dia da semana a empresa vende mais?
- › Em que horário?
- › É interessante colocar mais pessoas para atender em um horário diferente do que se imaginava antes?
- › Existem oscilações dentro do ano que “sempre acontecem”?

São muitas possibilidades de insights!



## Nomenclatura

SQL não é uma coisa única. Existem MySQL, PostgreSQL, Microsoft SQL Server etc. Ainda que os comandos básicos sejam os mesmos, esse não é o caso para todos os comandos.

**A nomenclatura das funções relacionadas a dados varia muito.** Aqui, falaremos das disponíveis no SQLITE, mas, no “Quero mais”, você encontra mais informações de sintaxe para as 7 diferentes formas que o SQL pode apresentar.

## Comandos

### CURRENT\_DATE

retorna como resultado a data atual.

### DATE()

retorna como resultado uma data no formato AAAA-MM-DD.

### STRFTIME

retorna como resultado a data formatada de acordo com as especificações.



# Mão na massa

- ▶ Quantas vendas foram feitas no mês passado?  
Qual foi o valor total?
- ▶ Qual é a evolução de vendas mês a mês desde que começamos nossas operações?

# Pulo do gato

Use variáveis, não o valor “na pedra”.

Se você monta uma consulta com o período com variáveis, você consegue fazer a mesma consulta e atualizar o seu número apertando apenas um botão.

Isso garante mais eficiência, sem perda de tempo trocando datas manualmente, além de possibilitar resultados de análises **mais confiáveis**, sem chance de erro humano.

## Desafio Conquer

Quais são as 8 subcategorias que venderam mais de 10 e menos de 100 produtos em 2022? Quanto eles somam em valor de venda?

Crie uma marcação para “de 10 a 50 produtos vendidos” e outra “de 50 a 100 produtos vendidos”.



## Quero mais:



**Aprendendo SQL:**  
Dominando os  
Fundamentos de  
SQL



Tech on the net



## Anotações

