

SQL

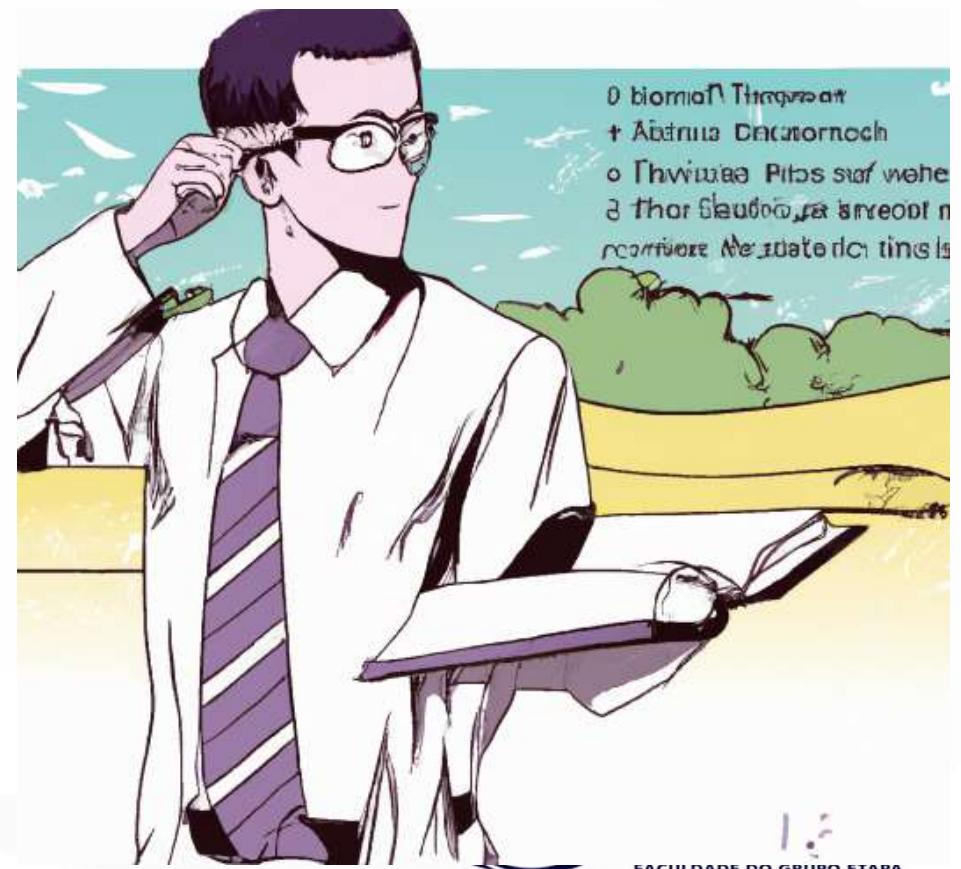
POR QUE SQL É IMPORTANTE?

SQL é a linguagem que processa dados

Vivemos na **era da informação**;

Toda empresa que busca sobreviver neste ambiente precisa **armazenar e saber trabalhar com os dados** que seu negócio gera;

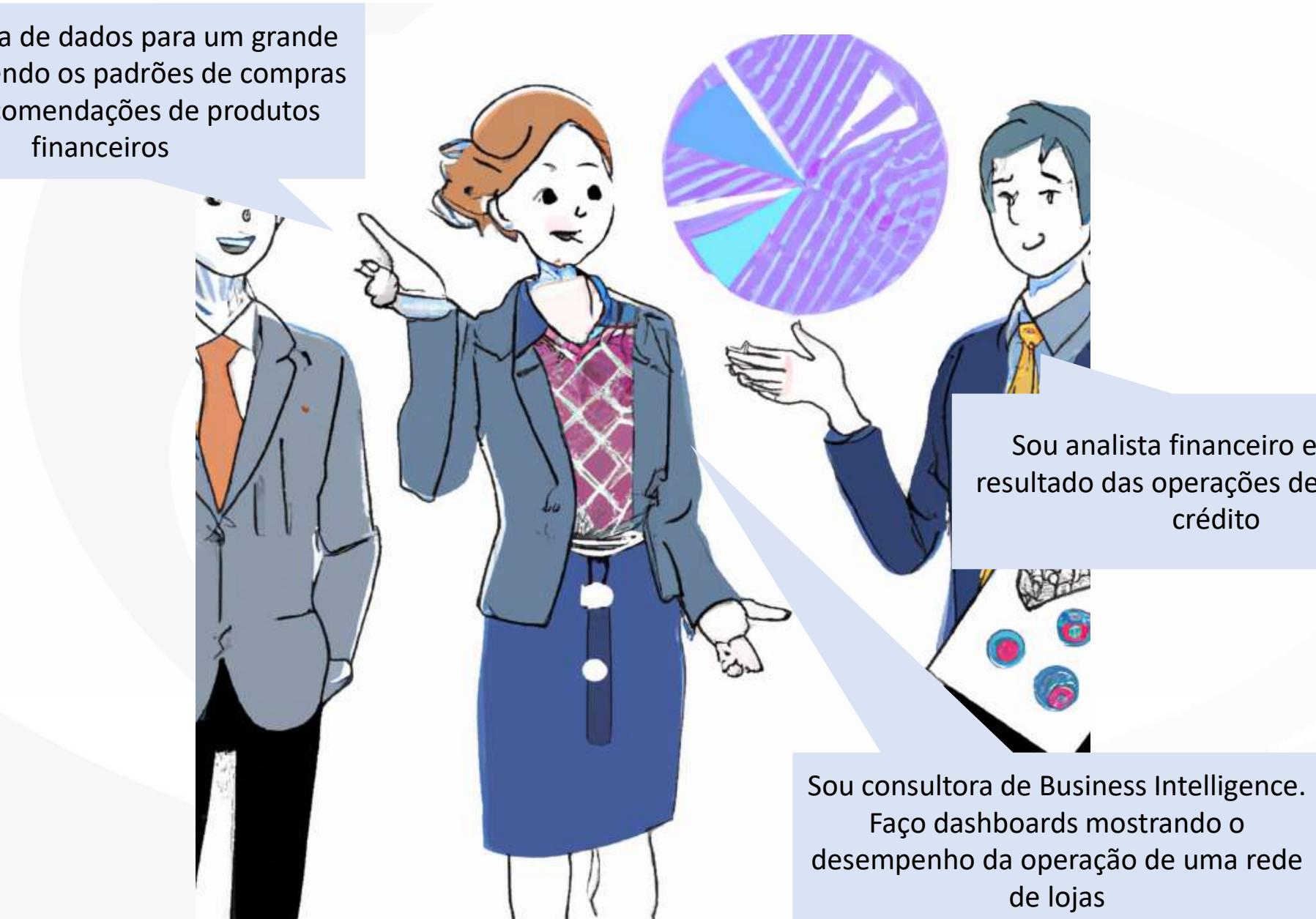
Bancos, varejistas, seguradoras e inúmeras empresas já vivem desta maneira; E com isso a **demandar por profissionais** que saiba tratar e manipular dados **só aumenta**;



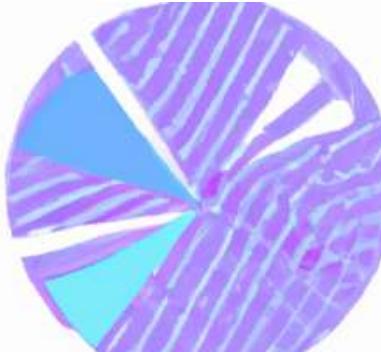


Pai, é verdade que todos estes itens à venda estão armazenados num banco de dados?

Sim filho todos os itens estão num enorme banco de dados e há pessoas analisando o padrão das nossas compras



Sou analista de dados para um grande banco. Entendo os padrões de compras e faço recomendações de produtos financeiros



Sou analista financeiro e avalio o resultado das operações de cartões de crédito

Sou consultora de Business Intelligence. Faço dashboards mostrando o desempenho da operação de uma rede de lojas

Minha história com SQL

Precisamos fazer data-mining dessa base de varejo;
Ela contém as vendas do último ano de todos os itens das nossas 400 lojas;
Você acha que é possível?

Sem problema, chefe! Entendo tudo de Excel



Minha história com SQL

Tem certeza que você vai conseguir
tratar todas aquelas informações só no
Excel?



Naquela época por não saber SQL tive muitas dúvidas



Existe uma maneira fácil de tratar esses dados?

Teria como fazer isso no meu laptop que tem pouca memória?

Existem ferramentas que podem me ajudar?

Com SQL aprendi que posso fazer tratamentos de base complexos;

E me desenvolvi trabalhando na prática diariamente por vários anos!

Você não terá que passar pelos mesmos problemas que eu passei porque não sabia disso na época.

Então, apresento a você um guia descomplicado de como usar o SQL.



LJL
FACULDADE DO GRUPO ETAPA

Vagas anunciadas recentemente e que precisam de SQL

Machine Learning Engineer

Randstad USA - Atlanta, GA • Temporarily remote

⚡ Responded to 75% or more applications in the past 30 days, typically within 1 day.

[Apply Now](#)



- Expert level proficiency with Python and libraries for machine learning such as scikit-learn and pandas
- Talend or Kapow
- Predictive modeling
- Cloud platforms such as BigQuery and Snowflake
- Expert-level SQL
- Knowledge of data cleaning, wrangling, visualization, and reporting, with an understanding of the best, most efficient use of associated tools and applications to complete these tasks.
- Deep knowledge of data mining, machine learning, natural language processing, or information retrieval.
- Functional Competencies
- Experience working closely with and supporting analytic, statistical, or data science functions

Machine Learning Engineer

BetterUp ★★★★☆ 3 reviews - California • Remote

[Apply Now](#)



- 5+ years of relevant experience, at least part of which in a startup environment
- Alignment with BetterUp mission of enabling behavior change
- Succeeded in a remote work environment
- Strong verbal and written communication
- Impressive portfolio (patents, publications, Kaggle profile, etc)
- Advanced Python expertise with data analysis and machine learning libraries (e.g. Numpy, tensorflow, etc)
- Experience with advanced machine learning architectures and models (e.g. NLP, Deep learning, etc)
- Effective analytical presentation skills using Jupyter notebooks and static presentation formats
- Efficient database modeling and cleaning techniques (SQL)
- Experience with cloud computing techniques (AWS, GCP or Azure)

CONFIGURAÇÃO DA FERRAMENTA

Além do SQLite existem outros fornecedores de bancos de dados
(MySQL, Postgres, SQL Server, etc)

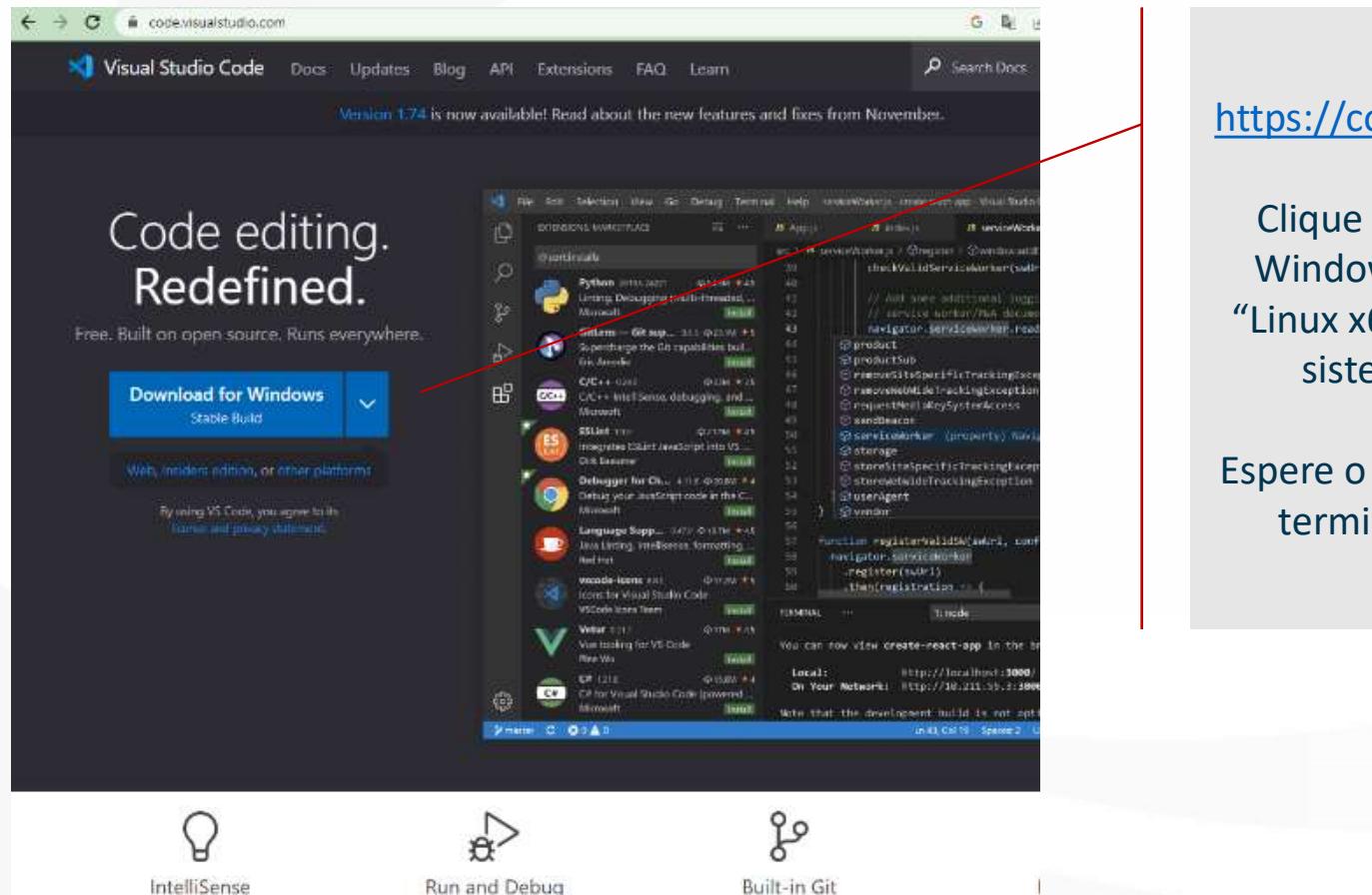
Usaremos o SQLite por ser simples, fácil de usar e já ter as ferramentas
necessárias ao aprendizado

Primeiro passo: Fazer download de um ou mais arquivos .db

Utilize o link a seguir para baixar qualquer um dos arquivos “.db” disponíveis:

<https://drive.google.com/drive/u/0/folders/1Ng9mlbGSnEhrCSkcUz89qbTTR8C4HYj8>

Segundo passo: Instalar o Visual Studio Code caso não o tenha

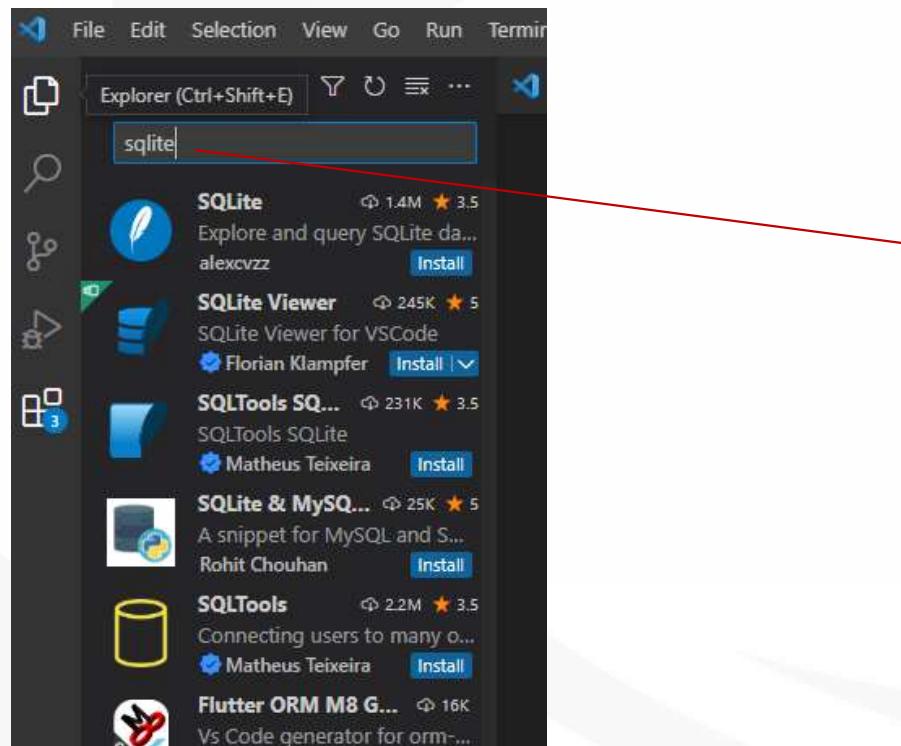


Entre em:
<https://code.visualstudio.com/>

Clique em: “Download for Windows” ou “macOS” ou “Linux x64”. (A versão do seu sistema operacional)

Espere o download terminar e termine a instalação do aplicativo

Terceiro passo: Instalar a extensão do SQLite para o Visual Studio Code



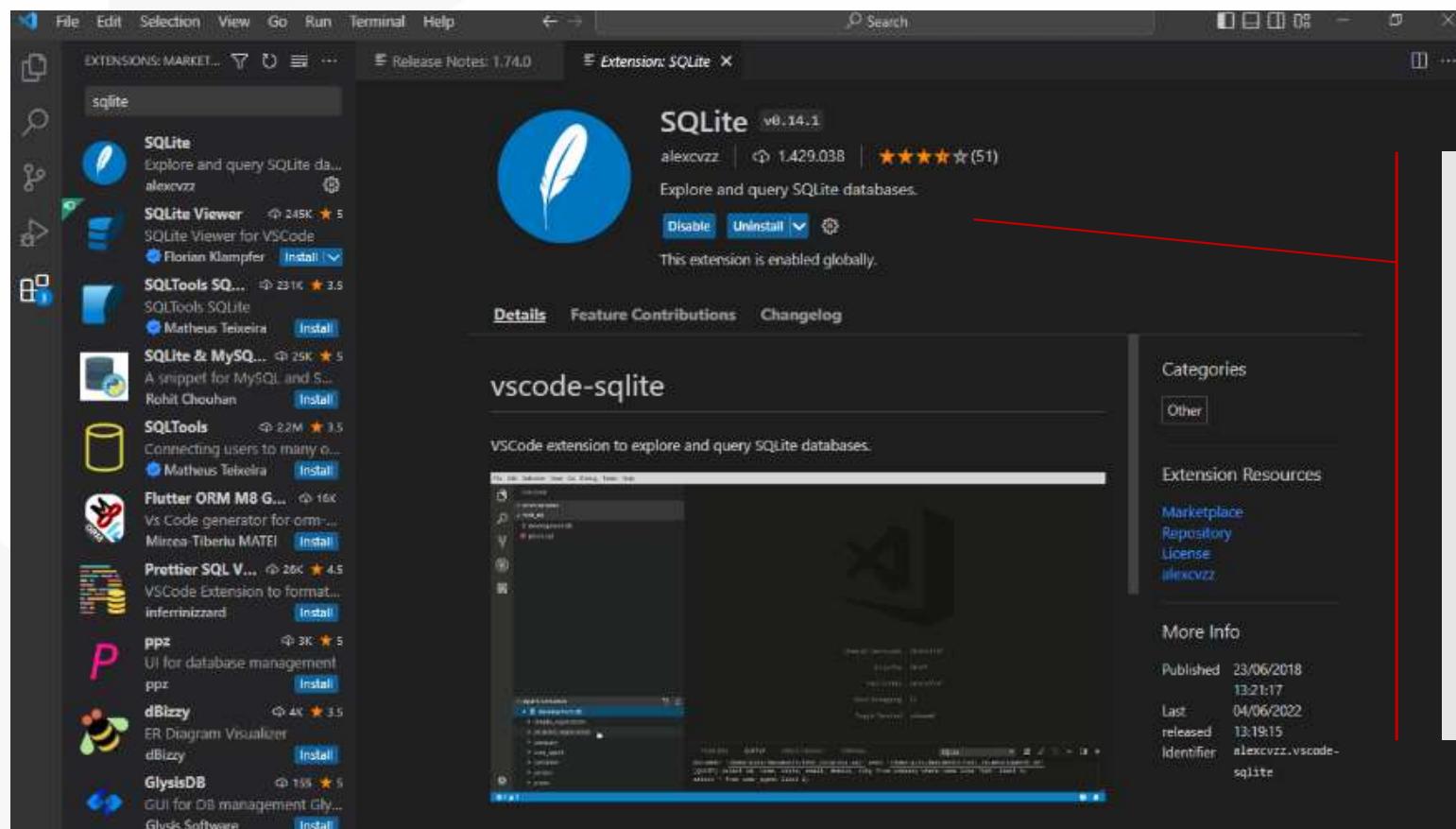
Uma vez aberto o Visual Studio Code, clique na lupa no lado esquerdo da tela.

Dentro da caixinha de busca escreva: “sqlite”

Espere ele trazer os resultados da busca

Instale aquela que tiver o nome “alexcvzz” no seu descriptivo

Terceiro passo: Instalar a extensão do SQLite para o Visual Studio Code



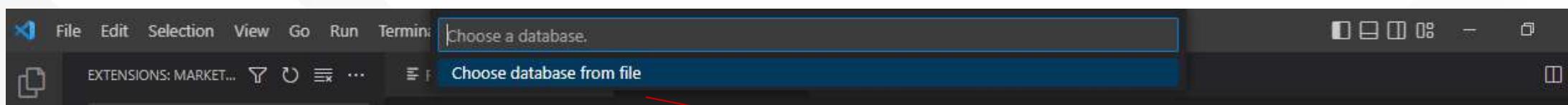
Esta é a tela que mostra que a extensão foi instalada

Quarto passo: Conectar o Visual Studio Code a algum banco de dados



No canto superior da tela, no campo de busca, escreva: ">sqlite". O ">" é importante!

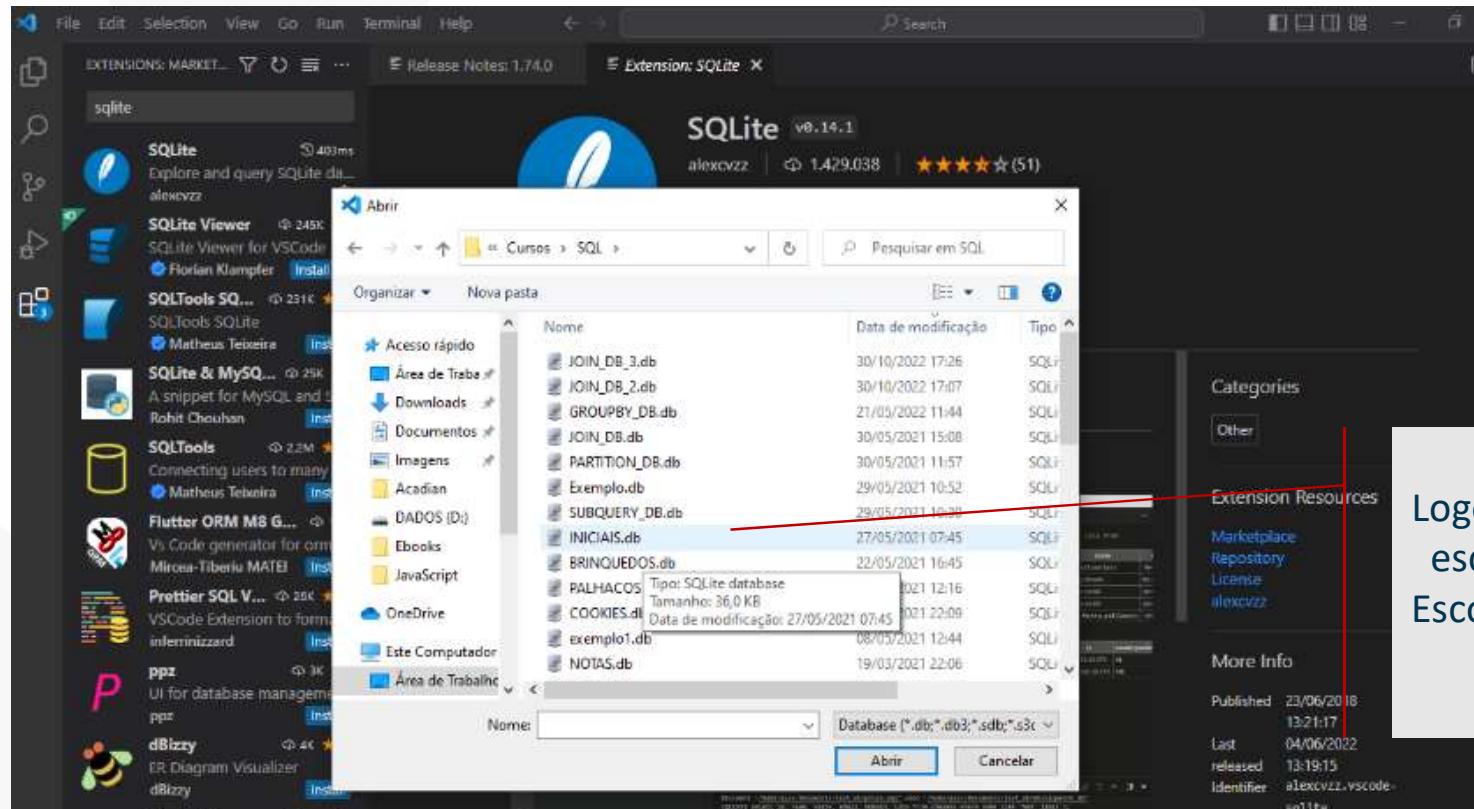
Seleciona "SQLite: Open Database"



Logo após surgirá esta opção:
"Choose database from file".

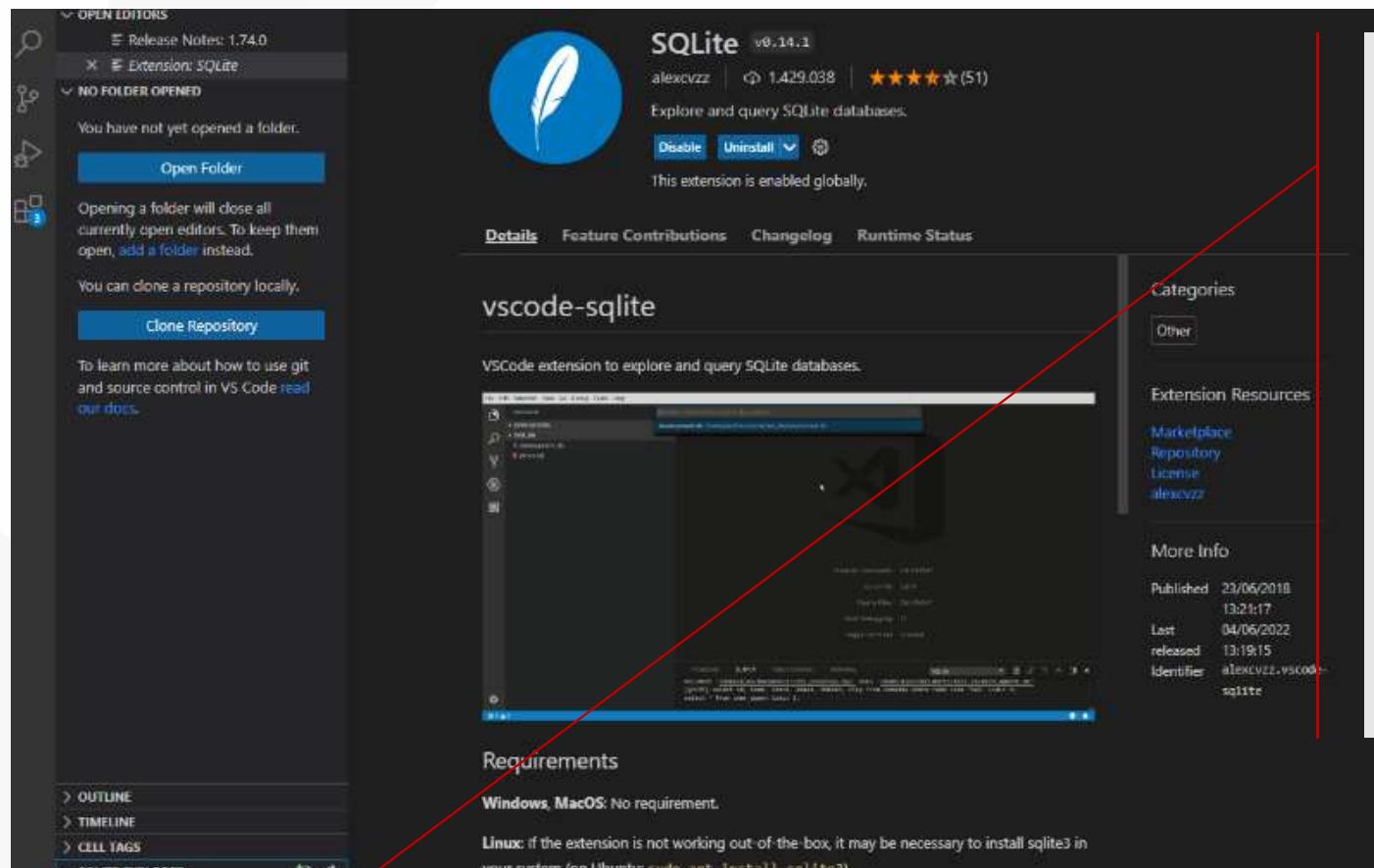
Seleciona ela. Clicando sobre

Quarto passo: Conectar o Visual Studio Code a algum banco de dados



Logo após surgirá esta tela para escolher algum arquivo ".db". Escolhe o "INICIAIS.db" e clique em "Abrir"

Quinto passo: Selecionar o banco de dados a ser trabalhado



Uma vez instalada a extensão do SQLite e aberto um banco de dados, surgirá este menu: “SQLITE EXPLORER”.

Dentro dele temos o banco de dados que você acabou de inserir: o “INICIAIS.db”

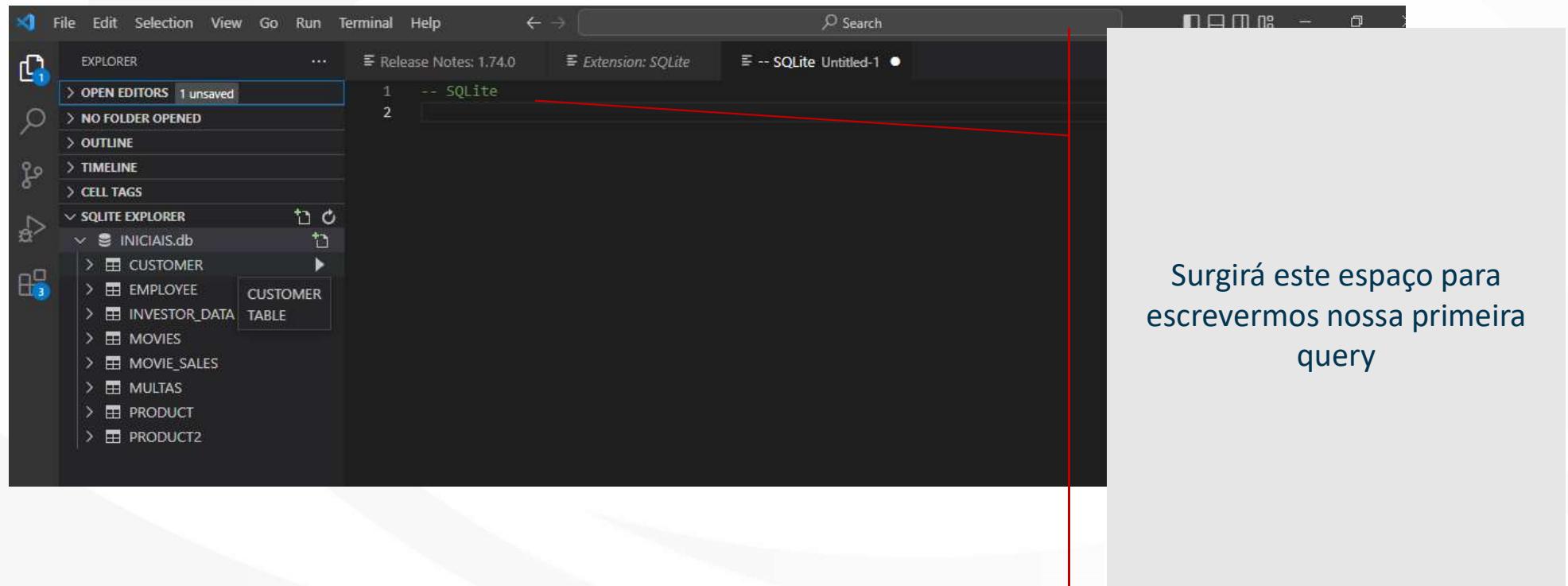
Clique no ícone que parece uma folha de papel com um mais em cima ao lado do “INICIAIS.db”

Queries são comandos para fazer consultas nos bancos de dados.

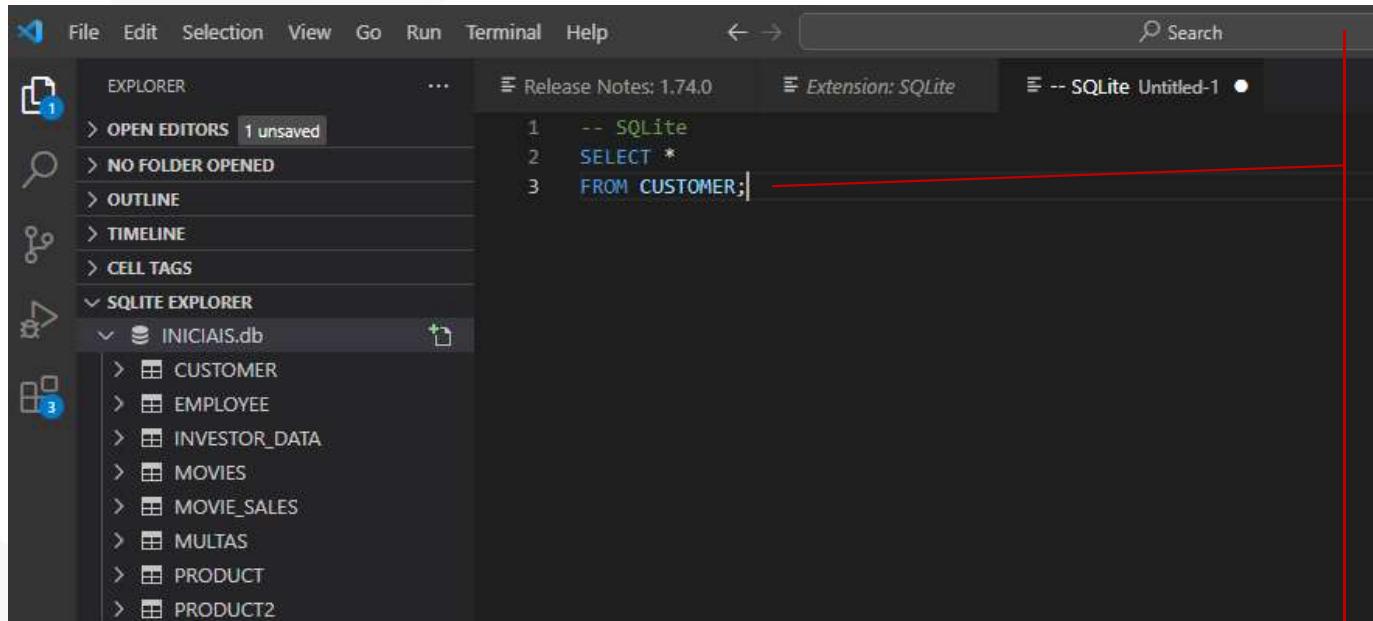
E é o que normalmente escrevemos quando estamos programando em SQL.



Sexto passo: Escrever a sua primeira “query”



Sexto passo: Escrever a sua primeira “query”



A screenshot of a dark-themed SQLite editor interface. The top bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar has sections for OPEN EDITORS (1 unsaved), NO FOLDER OPENED, OUTLINE, TIMELINE, CELL TAGS, and SQLITE EXPLORER, which lists tables like CUSTOMER, EMPLOYEE, INVESTOR_DATA, MOVIES, MOVIE_SALES, MULTAS, PRODUCT, and PRODUCT2. The main area shows a code editor with the following SQL query:

```
1 -- SQLite
2 SELECT *
3 FROM CUSTOMER;
```

A red vertical line highlights the third line of the query. To the right of the editor is a light gray panel containing instructions:

Escreva **SELECT * FROM CUSTOMER;**

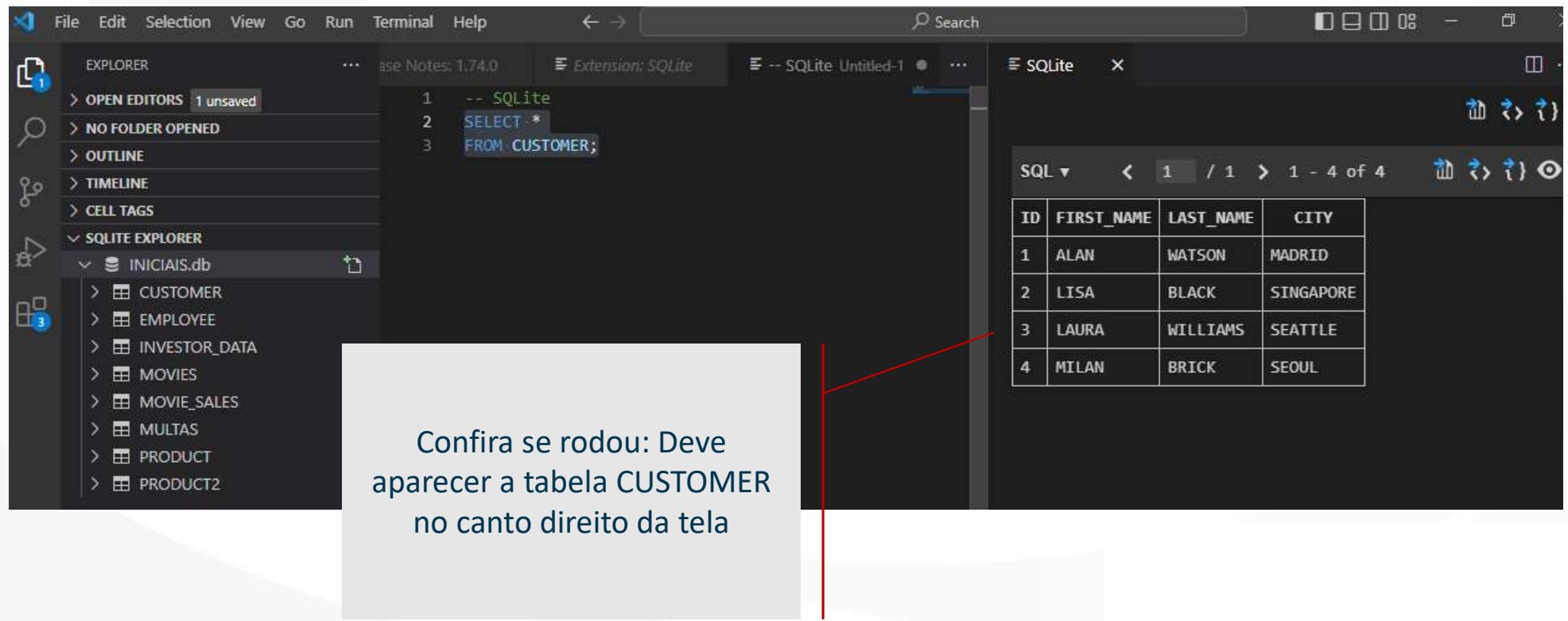
Selecione o que você acabou de escrever com o mouse e digite: **Ctrl+Shift+q**



Para rodar um comando SQL primeiro selecionamos a query e depois pedimos para rodar.

A ferramenta só vai rodar o comando selecionado, e nada mais.

Sétimo passo: Conferir o resultado da query



The screenshot shows the SQLite Database Browser interface. On the left, the Explorer sidebar lists databases and tables. In the center, a SQL editor window contains the following query:

```
-- SQLite
SELECT *
FROM CUSTOMER;
```

To the right, a results grid displays the data from the CUSTOMER table:

ID	FIRST_NAME	LAST_NAME	CITY
1	ALAN	WATSON	MADRID
2	LISA	BLACK	SINGAPORE
3	LAURA	WILLIAMS	SEATTLE
4	MILAN	BRICK	SEOUL

A red callout box with a red arrow points from the text "Confira se rodou: Deve aparecer a tabela CUSTOMER no canto direito da tela" to the results grid.

Confira se rodou: Deve aparecer a tabela CUSTOMER no canto direito da tela

As próximas páginas são opcionais caso você já tenha instalado o Visual Studio Code e rodado sua primeira query com sucesso.

Elas mostram uma alternativa ao Visual Studio Code.

O SQLite Studio.

Primeiro passo: Fazer download do SQLite Studio em: <https://sqlitestudio.pl/>

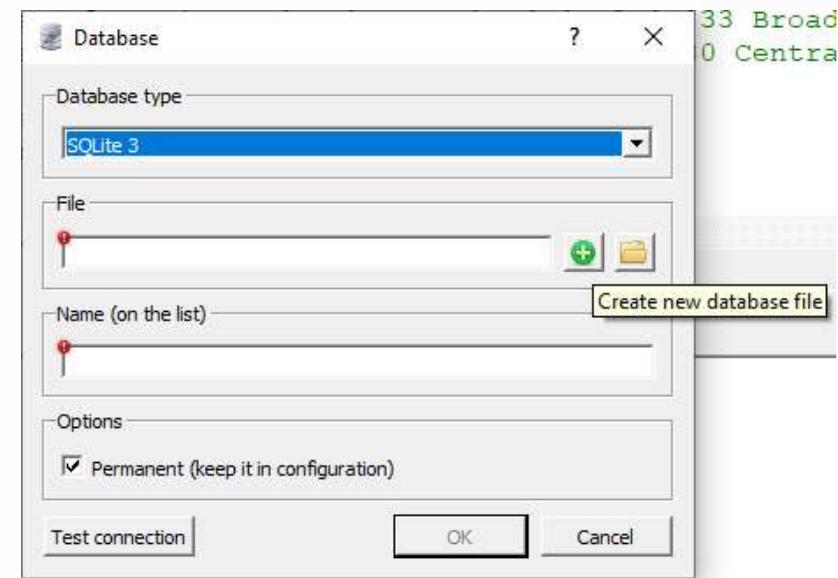


Segundo passo: Adicionar um banco de dados

1. No SqliteStudio vá em

Database > Add a DataBase > File > Create New Database File (Botão Mais)

2. Crie um arquivo banco de dados em algum lugar no seu computador.

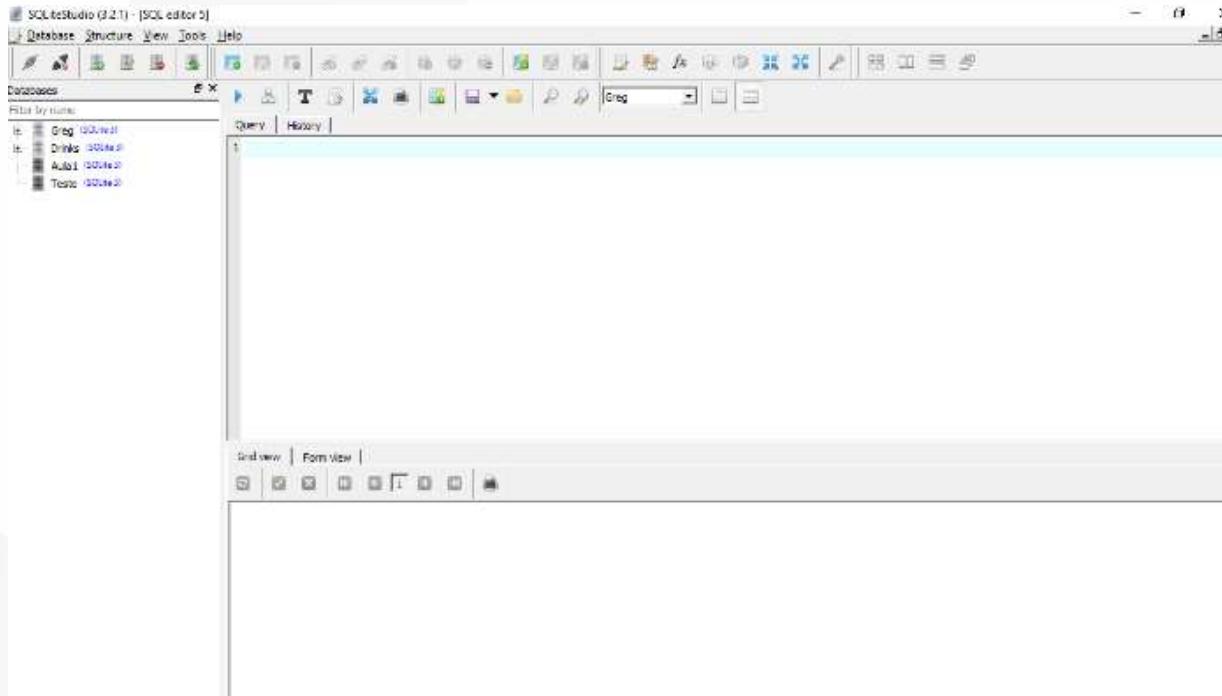


Terceiro passo: Criar um ambiente para escrever queries

Vá em: Open SQL Editor ou (Alt + E)



Ambiente de Desenvolvimento do SQLStudio



Queries são comandos para fazer consultas nos bancos de dados.

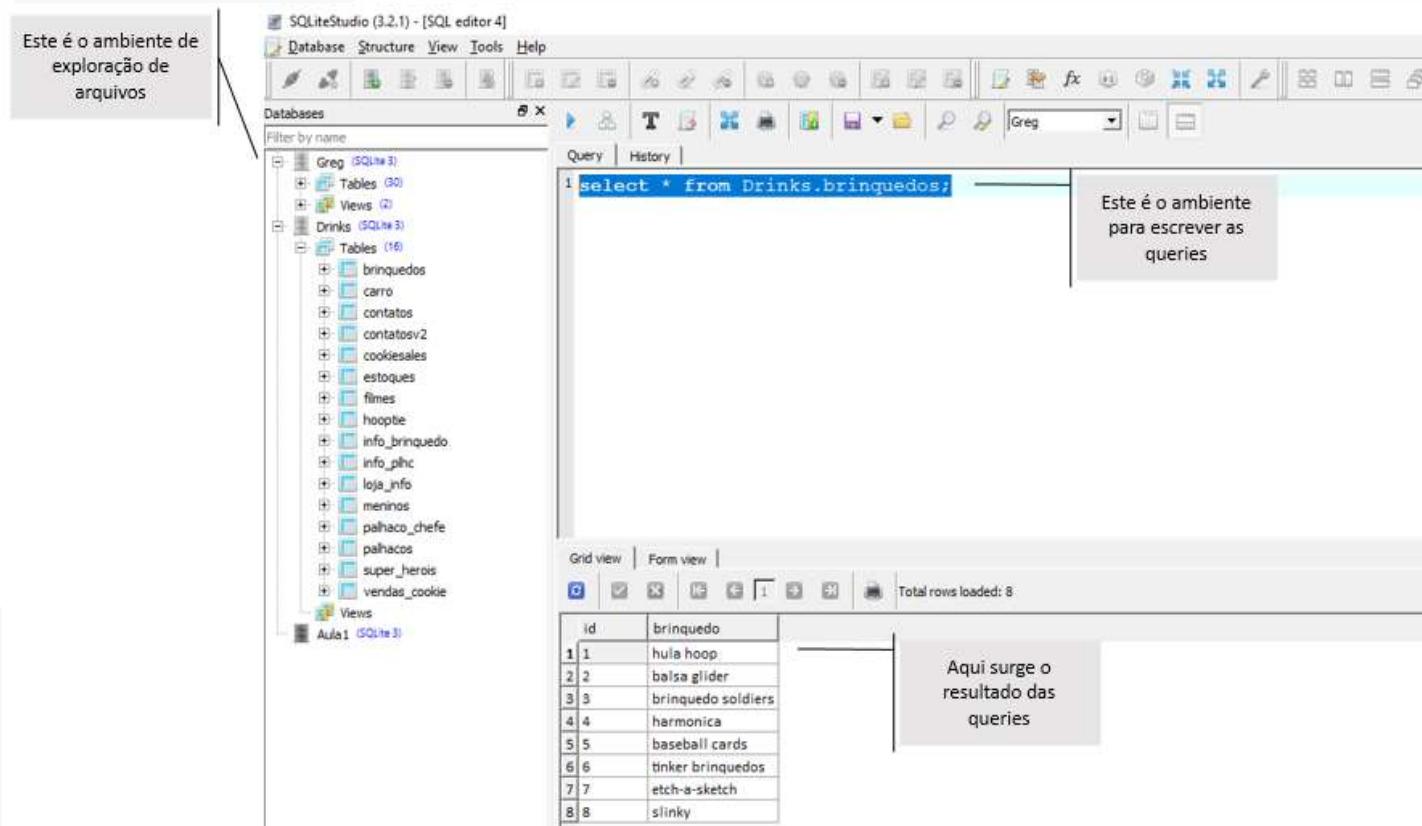
E é o que normalmente escrevemos quando estamos programando em SQL.



Terceiro passo: Escreva a sua query e aperte no botão de “play”

SELECT 'Hello World!'

Terceiro passo: Escreva a sua query e aperte no botão de “play”





Para rodar um comando SQL primeiro selecionamos a query e depois pedimos para rodar.

A ferramenta só vai rodar o comando selecionado, e nada mais.

Primeiras consultas no banco de dados

Encontre o que precisa

No trabalho geralmente usamos tabelas para encontrar respostas de negócios

Saber filtrar é uma habilidade muito útil no dia-a-dia de quem trabalha com tabelas

E ao filtrar trazemos apenas as informações que precisamos

Então vamos lá:

- Nossa primeira tarefa em SQL é aprender a fazer **filtros!**
- E para fazer filtros usamos o operador **WHERE**





Hummm... antes de escrever SQL
preciso entender muito bem a
tabela que estou trabalhando

Vamos conhecer a tabela de clientes

id_cliente	nome	sobrenome	cidade	país	data_nasc	pet
1	Alice	Santos	Rio de Janeiro	Brasil	29/12/1976	NULL
2	Miguel	Oliveira	São Paulo	Brasil	20/12/1986	gato
3	Artur	Souza	Rio de Janeiro	Brasil	19/05/1983	cachorro
4	Bernardo	Rodrigues	Nova York	EUA	13/01/1977	cachorro
5	Helena	Ferreira	São Paulo	Brasil	16/06/1989	NULL
6	Sophia	Alves	Londres	Inglaterra	06/10/2005	NULL
7	Isabella	Pereira	Berlin	Alemanha	30/04/1970	cachorro
8	Heloisa	Gomes	Rio de Janeiro	Brasil	05/06/2008	gato
9	Lorena	Ribeiro	Buenos Aires	Argentina	04/07/1998	gato
10	Nicola	Lima	Vancouver	Canada	06/02/2009	gato

Você conseguiria descrever quais informações temos nesta tabela?

Como filtrar apenas os clientes que são da cidade BERLIN?

Como filtrar apenas os clientes que são da cidade BERLIN?

```
SELECT * FROM CLIENTES WHERE CIDADE = 'Berlin';
```

Partimos de uma consulta na tabela clientes e aplicamos o filtro de cidade ='Berlin'

* Dica:

- > Como queríamos filtrar apenas 1 item, usamos o sinal de =
 - > Já para diversos itens, geralmente utilizamos “in” [se queremos apenas o que está no filtro] ou “not in” [se queremos tudo menos o que está no filtro]
- Ex: *select * from clientes where cidade in ('Berlin', 'Londres', 'Nova York')*



Por que você colocou
Berlin entre aspas
simples?

Para que o computador
reconheça algo como
texto, ou seja, string, é
preciso colocar a
palavra entre aspas.
Logo, temos 'Berlin'

Repare que como cidade é um campo do tipo string/varchar, é preciso traduzir Berlin em uma string.

Então temos:

```
SELECT * FROM customers  
WHERE city = 'Berlin';
```

E não:

```
SELECT * FROM customers  
WHERE city = Berlin;
```





SQL não diferencia maiúsculas de minúsculas

Então SQL NÃO é caso-sensitivo



SQL não é caso-sensitivo, então tanto faz escrever:

select ou SELECT

Mas normalmente para as palavras-chave como SELECT, FROM, WHERE usamos caixa-alta



Mas atenção!
Estamos falando apenas para comandos SQL e nomes das colunas.

Já para os conteúdos de dentro da célula da tabela, é caso sensível. Ou seja:

Where tanto faz ser WHERE ou where (visto que é um comando).

Já no caso de where city = 'Berlin' (não posso usar 'BERLIN', ou mesmo 'berlin'... a correspondência precisa ser exatamente igual ao que estamos filtrando da informação da célula (situação parecida com o que acontece no Excel)).

Como filtrar apenas os clientes cuja cidade não é BERLIN?

id_cliente	nome	sobrenome	cidade	país	data_nasc	pet
1	Alice	Santos	Rio de Janeiro	Brasil	29/12/1976	NULL
2	Miguel	Oliveira	São Paulo	Brasil	20/12/1986	gato
3	Artur	Souza	Rio de Janeiro	Brasil	19/05/1983	cachorro
4	Bernardo	Rodrigues	Nova York	EUA	13/01/1977	cachorro
5	Helena	Ferreira	São Paulo	Brasil	16/06/1989	NULL
6	Sophia	Alves	Londres	Inglaterra	06/10/2005	NULL
7	Isabella	Pereira	Berlin	Alemanha	30/04/1970	cachorro
8	Heloisa	Gomes	Rio de Janeiro	Brasil	05/06/2008	gato
9	Lorena	Ribeiro	Buenos Aires	Argentina	04/07/1998	gato
10	Nicola	Lima	Vancouver	Canada	06/02/2009	gato

Como filtrar apenas os clientes cuja cidade não é BERLIN?

```
SELECT * FROM CLIENTES WHERE CIDADE <> 'Berlin';
```

Ou:

```
SELECT * FROM CLIENTES WHERE CIDADE != 'Berlin';
```

Como filtrar os clientes que nasceram depois de 01/01/1984?

id_cliente	nome	sobrenome	cidade	país	data_nasc	pet
1	Alice	Santos	Rio de Janeiro	Brasil	29/12/1976	NULL
2	Miguel	Oliveira	São Paulo	Brasil	20/12/1986	gato
3	Artur	Souza	Rio de Janeiro	Brasil	19/05/1983	cachorro
4	Bernardo	Rodrigues	Nova York	EUA	13/01/1977	cachorro
5	Helena	Ferreira	São Paulo	Brasil	16/06/1989	NULL
6	Sophia	Alves	Londres	Inglaterra	06/10/2005	NULL
7	Isabella	Pereira	Berlin	Alemanha	30/04/1970	cachorro
8	Heloisa	Gomes	Rio de Janeiro	Brasil	05/06/2008	gato
9	Lorena	Ribeiro	Buenos Aires	Argentina	04/07/1998	gato
10	Nicola	Lima	Vancouver	Canada	06/02/2009	gato

Como filtrar os clientes que nasceram depois de 01/01/1984?

```
SELECT * FROM CLIENTES WHERE DATA_NASC > '1984-01-01';
```

Obs: > seleciona apenas datas depois.
Quando precisar selecionar também a data que aparece na função, se usa >= (ou seja, que seja maior ou igual a data selecionada)



Normalmente a data é no formato YYYY-DD-MM

YYYY (ano ex: 1994)

DD (dia ex: 04)

MM (mês ex: 12)

E também na data precisamos colocar aspas simples

Como filtrar os clientes que tem pet= 'cachorro' OU a cidade = 'Berlin'?

id_cliente	nome	sobrenome	cidade	país	data_nasc	pet
1	Alice	Santos	Rio de Janeiro	Brasil	29/12/1976	NULL
2	Miguel	Oliveira	São Paulo	Brasil	20/12/1986	gato
3	Artur	Souza	Rio de Janeiro	Brasil	19/05/1983	cachorro
4	Bernardo	Rodrigues	Nova York	EUA	13/01/1977	cachorro
5	Helena	Ferreira	São Paulo	Brasil	16/06/1989	NULL
6	Sophia	Alves	Londres	Inglaterra	06/10/2005	NULL
7	Isabella	Pereira	Berlin	Alemanha	30/04/1970	cachorro
8	Heloisa	Gomes	Rio de Janeiro	Brasil	05/06/2008	gato
9	Lorena	Ribeiro	Buenos Aires	Argentina	04/07/1998	gato
10	Nicola	Lima	Vancouver	Canada	06/02/2009	gato

Como filtrar os clientes que tem pet= 'cachorro' OU a cidade = 'Berlin'?

```
SELECT * FROM CLIENTES  
WHERE PET = 'cachorro' OR CIDADE = 'Berlin';
```

Se caso quisermos selecionar clientes que tenham cachorro e também seja da cidade Berlin, usamos a função **AND** na posição onde está a função **OR**

Dada a tabela abaixo, escrever uma query que traga o filme com id = 6

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que traga o filme com id = 6

```
SELECT * FROM FILMES WHERE ID_FILME = 6;
```

Dada a tabela abaixo, escrever uma query que traga os filmes lançados entre 2000 e 2010

(incluindo 2000 e 2010)

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que traga os filmes lançados entre 2000 e 2010
(incluindo 2000 e 2010)

```
SELECT * FROM FILMES WHERE ANO BETWEEN 2000 AND 2010;
```

Dada a tabela abaixo, escrever uma query que traga os filmes não lançados entre 2000 e 2010

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefao	Francis Ford Copolla	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	Davic Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que traga os filmes não lançados entre 2000 e 2010

```
SELECT * FROM FILMES WHERE ANO NOT BETWEEN 2000 AND 2010;
```

Dada a tabela abaixo, escrever uma query que traga os filmes cujo id seja de 1 até 5

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que traga os filmes cujo id seja de 1 até 5

```
SELECT * FROM FILMES WHERE ID_FILME <= 5;
```

Dada a tabela abaixo, escrever uma query que traga os filmes cujo id seja de 1 até 5 (mas apenas as colunas titulo e ano)

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que traga os filmes cujo id seja de 1 até 5 (mas apenas as colunas titulo e ano)

```
SELECT TITULO, ANO FROM FILMES WHERE ID_FILME <= 5;
```

Os próximos exercícios são baseados na tabela abaixo (world)

id_pais	nome	continente	populacao	pib
1	EUA	America	325.084.756	\$19.485.394.000.000
2	China	Asia	1.421.021.791	\$12.237.700.479.375
3	Japão	Asia	127.502.725	\$4.872.415.104.315
4	Alemanha	Europa	82.658.409	\$3.693.204.332.230
5	India	Asia	1.338.676.785	\$2.650.725.335.364
6	Reino Unido	Europa	66.727.461	\$2.637.866.340.434
7	França	Europa	64.842.509	\$2.582.501.307.216
8	Brasil	America	207.833.823	\$2.053.594.877.013
9	Italia	Europa	60.673.701	\$1.943.835.376.342
10	Canada	America	36.732.095	\$1.647.120.175.449

Você conseguiria descrever quais informações temos nesta tabela?

Qual código gera uma lista apenas com o nome dos países que tem população maior que 1.000.000.000?

- A)

```
SELECT
    nome
FROM populacao
WHERE populacao > 1000000000
```
- B)

```
SELECT
    nome, populacao
FROM world
WHERE populacao > 1000000000
```
- C)

```
SELECT
    nome, populacao
FROM world
WHERE populacao < 1000000000
```

- D)

```
SELECT
    nome
FROM world
WHERE populacao > 1000000000
```
- E)

```
SELECT
    populacao
FROM nome
WHERE populacao > 1000000000
```

Resposta: Qual código gera uma lista apenas com o nome dos países que tem população maior que 1.000.000.000?

- A)

```
SELECT
    nome
FROM populacao
WHERE populacao > 1000000000
```
- B)

```
SELECT
    nome, populacao
FROM world
WHERE populacao > 1000000000
```
- C)

```
SELECT
    nome, populacao
FROM world
WHERE populacao < 1000000000
```

- D)

```
SELECT
    nome
FROM world
WHERE populacao > 1000000000
```
- E)

```
SELECT
    populacao
FROM nome
WHERE populacao > 1000000000
```

Qual código produz a tabela abaixo?

a)
FROM world
SELECT nome, populacao
WHERE populacao BETWEEN 30000000 and 70000000

b)
FROM nome, populacao
WHERE populacao BETWEEN 30000000 and 70000000
SELECT world

c)
SELECT nome, populacao
FROM world
WHERE populacao BETWEEN 30000000 and 70000000

d)
SELECT populacao BETWEEN 30000000 and 70000000
FROM world

e)
WHERE populacao BETWEEN 30000000 and 70000000
SELECT nome, populacao
FROM world

nome	populacao
Reino Unido	66.727.461
França	64.842.509
Italia	60.673.701
Canada	36.732.095

Resposta: Qual código produz a tabela abaixo?

```
a)  
FROM world  
SELECT nome, populacao  
WHERE populacao BETWEEN 30000000 and 70000000  
  
b)  
FROM nome, populacao  
WHERE populacao BETWEEN 30000000 and 70000000  
SELECT world  
  
c)  
SELECT nome, populacao  
FROM world  
WHERE populacao BETWEEN 30000000 and 70000000  
  
d)  
SELECT populacao BETWEEN 30000000 and 70000000  
FROM world  
  
e)  
WHERE populacao BETWEEN 30000000 and 70000000  
SELECT nome, populacao  
FROM world
```

nome	populacao
Reino Unido	66.727.461
França	64.842.509
Italia	60.673.701
Canada	36.732.095

Em qual situação você listaria os nomes das colunas no lugar de usar um asterisco ao fazer um SELECT?

Em qual situação você listaria os nomes das colunas no lugar de usar um asterisco ao fazer um SELECT?

Resposta:

Quando você quer trazer apenas algumas colunas

Talvez por questões de confidencialidade, você não traria o salário de cada pessoa como uma resposta, por exemplo

Se fizermos BETWEEN 3 AND 5 a resposta inclui o 5?
E o 3?

Se fizermos BETWEEN 3 AND 5 a resposta inclui o 5?
E o 3?

Resposta:

Sim, o between é “bolinha fechada”, inclui o 3 e o 5

Qual a maneira de trazer todas as colunas de uma tabela?

Qual a maneira de trazer todas as colunas de uma tabela?

Resposta:

`SELECT * FROM TABELA`

Verdadeiro ou Falso.

Sempre que escrever um SELECT preciso escrever
WHERE



Verdadeiro ou Falso.

Sempre que escrever um SELECT preciso escrever WHERE

Resposta:

Falso

Verdadeiro ou Falso.
"NOT <" é equivalente a "<="

Verdadeiro ou Falso.

"NOT <" é equivalente a "<="

Resposta:

Falso

LIKE e WILDCARDS



Em muitos casos você pode não ser capaz de definir as condições exatas para fazer um filtro usando o operador '='.

E pode querer fazer **filtros com valores aproximados** às suas condições.

É para isso que serve o operador **LIKE**.

O operador LIKE permite identificar **strings (textos)** ‘**aproximadas**’, não necessariamente exatas.

Ele é usado **em conjunto com** operadores chamados **wildcards** ('%' e '_'), como veremos nos exemplos a seguir:



Como fazer uma query para trazer os clientes de cidades que começam com ‘S’?

id	nome	sobrenome	cidade
1	Alan	Williams	Seattle
2	Laura	Silva	São Paulo
3	Heloisa	Madureira	Fortaleza
4	Lorena	Ferreira	Manaus
5	Nicole	Pereira	Seul

Como fazer uma query para trazer os clientes de cidades que começam com ‘S’?

```
SELECT
  *
FROM CLIENTES
WHERE CIDADE LIKE 'S%';
```



id	nome	sobrenome	cidade
1	Alan	Williams	Seattle
2	Laura	Silva	São Paulo
3	Heloisa	Madureira	Fortaleza
4	Lorena	Ferreira	Manaus
5	Nicole	Pereira	Seul



A posição do wildcard % é importante.

Por exemplo:

Ex 1 - where cidade like 'S%' → não ter o % antes do S indica que antes do S não pode existir nada. Logo a string começa em S. E a presença do % após o S indica que na sequência o S pode estar acompanhado de qualquer complemento. Aqui teremos como resposta as cidades começadas em S (Seattle, São Paulo, Seul).

Ex 2 - where cidade like '%S%' → repare que agora posicionamos o wildcard % antes e depois do S. Logo, o resultado aqui muda, pois teremos cidades que podem possuir complemento tanto antes quanto depois do S. Então além das cidades mencionadas no ex1, também veríamos aqui a cidade Manaus.



Como o exercício pediu apenas clientes de cidades que começam com S, a resposta correta seria ***like 'S%'***, a qual posiciona o % apenas após o S.

O operador % traz qualquer caractere qualquer número de vezes.

Já o operador _ traz qualquer caractere mas apenas uma vez.

Assim a busca **like 'S%**' poderia trazer: São Paulo, Seattle e Seul.

Mas a busca **like 'S_'** só poderia trazer cidades (se existissem) como: 'Sa', 'Sb', 'Sc', ... 'Sz'.

Podemos ainda repetir o _ algumas vezes.

A busca **like 'S__'** traria 'Saa', 'Sab', 'Sbb', etc.



Como fazer uma query para trazer os clientes de cidades que tenham exatamente 4 caracteres?

id	nome	sobrenome	cidade
1	Alan	Williams	Seattle
2	Laura	Silva	São Paulo
3	Heloisa	Madureira	Fortaleza
4	Lorena	Ferreira	Manaus
5	Nicole	Pereira	Seul

Como fazer uma query para trazer os clientes de cidades que tenham exatamente 4 caracteres?

```
SELECT
  *
FROM CLIENTES
WHERE CIDADE LIKE '_ _ _';
```



id	nome	sobrenome	cidade
5	Nicole	Pereira	Seul



O wildcard "%" ajuda a buscar
"qualquer texto **qualquer número de**
vezes"

O operador "_" busca qualquer
caractere mas **apenas uma vez**



Se quiser filtrar qualquer caractere apenas uma vez posso usar: like '_'

Se quiser filtrar qualquer caractere exatamente duas vezes posso usar: like '__'

Se quiser filtrar qualquer caractere exatamente três vezes posso usar: like '___'

Como fazer uma query para trazer os clientes de cidades que começam com S e tenham ‘u’ na penúltima letra?

id	nome	sobrenome	cidade
1	Alan	Williams	Seattle
2	Laura	Silva	São Paulo
3	Heloisa	Madureira	Fortaleza
4	Lorena	Ferreira	Manaus
5	Nicole	Pereira	Seul

Resposta: Como fazer uma query para trazer os clientes de cidades que começam com S e tenham ‘u’ na penúltima letra?

```
SELECT
  *
FROM CLIENTES
WHERE CIDADE LIKE 'S%U_';
```

id	nome	sobrenome	cidade
5	Nicole	Pereira	Seul

O que significa colocar o “_” no final da seleção?

Significa que após o U podemos ter qualquer caractere mas apenas uma vez

Dada a tabela abaixo, escrever uma query que encontre todos os filmes que comecem com 'O'

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefao	Francis Ford Copolla	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	Davic Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que encontre todos os filmes que comecem com 'O'

```
SELECT
  *
FROM FILMES
WHERE TITULO LIKE 'O%';
```

Dada a tabela abaixo, escrever uma query que encontre todos os filmes dirigidos por algum Steven

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Copolla	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que encontre todos os filmes dirigidos por algum Steven

```
SELECT
  *
FROM FILMES
WHERE DIRETOR LIKE '%Steven%';
```

Dada a tabela abaixo, escrever uma query que encontre todos os filmes não dirigidos por algum Steven

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefao	Francis Ford Copolla	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	Davic Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Dada a tabela abaixo, escrever uma query que encontre todos os filmes não dirigidos por algum Steven

```
SELECT
  *
FROM FILMES
WHERE DIRETOR NOT LIKE '%Steven%';
```

Os próximos exercícios são baseados na tabela abaixo (world)

id_pais	nome	continente	populacao	pib
1	EUA	America	325.084.756	\$19.485.394.000.000
2	China	Asia	1.421.021.791	\$12.237.700.479.375
3	Japão	Asia	127.502.725	\$4.872.415.104.315
4	Alemanha	Europa	82.658.409	\$3.693.204.332.230
5	India	Asia	1.338.676.785	\$2.650.725.335.364
6	Reino Unido	Europa	66.727.461	\$2.637.866.340.434
7	França	Europa	64.842.509	\$2.582.501.307.216
8	Brasil	America	207.833.823	\$2.053.594.877.013
9	Italia	Europa	60.673.701	\$1.943.835.376.342
10	Canada	America	36.732.095	\$1.647.120.175.449

Qual o resultado da query abaixo?

```
SELECT nome, continente, populacao  
FROM world  
WHERE continente like 'A%'
```

A)

nome	continente	populacao
EUA	America	\$325.084.756
China	Asia	\$1.421.021.791
Japão	Asia	\$127.502.725
India	Asia	\$1.338.676.785
Brasil	America	\$207.833.823
Canada	America	\$36.732.095

B)

nome	populacao	continente
EUA	\$325.084.756	America
China	\$1.421.021.791	Asia
Japão	\$127.502.725	Asia
India	\$1.338.676.785	Asia
Brasil	\$207.833.823	America
Canada	\$36.732.095	America

C)

id_pais	nome	continente	populacao	pib
1	EUA	America	325.084.756	\$19.485.394.000.000
2	China	Asia	1.421.021.791	\$12.237.700.479.375
3	Japão	Asia	127.502.725	\$4.872.415.104.315
8	Brasil	America	207.833.823	\$2.053.594.877.013
5	India	Asia	1.338.676.785	\$2.650.725.335.364
10	Canada	America	36.732.095	\$1.647.120.175.449

Resposta: Qual o resultado da query abaixo?

```
SELECT nome, continente, populacao  
FROM world  
WHERE continente like 'A%'
```

A)	nome	continente	populacao
EUA	America	\$325.084.756	
China	Asia	\$1.421.021.791	
Japão	Asia	\$127.502.725	
India	Asia	\$1.338.676.785	
Brasil	America	\$207.833.823	
Canada	America	\$36.732.095	

B)	nome	populacao	continente
EUA	\$325.084.756	America	
China	\$1.421.021.791	Asia	
Japão	\$127.502.725	Asia	
India	\$1.338.676.785	Asia	
Brasil	\$207.833.823	America	
Canada	\$36.732.095	America	

C)	id_pais	nome	continente	populacao	pib
	1	EUA	America	325.084.756	\$19.485.394.000.000
	2	China	Asia	1.421.021.791	\$12.237.700.479.375
	3	Japão	Asia	127.502.725	\$4.872.415.104.315
	8	Brasil	America	207.833.823	\$2.053.594.877.013
	5	India	Asia	1.338.676.785	\$2.650.725.335.364
	10	Canada	America	36.732.095	\$1.647.120.175.449

Qual query gera uma lista de países que começam com A ou I?

A)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' AND nome LIKE 'I%'
```

B)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' OR 'I%'
```

C)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' OR LIKE 'I%'
```

D)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' OR nome LIKE 'I%'
```

E)

```
SELECT
    world
FROM nome
WHERE nome LIKE 'A%' OR nome LIKE 'I%'
```

Resposta: Qual query gera uma lista de países que começam com A ou I?

A)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' AND nome LIKE 'I%'
```

B)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' OR 'I%'
```

C)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' OR LIKE 'I%'
```

D)

```
SELECT
    nome
FROM world
WHERE nome LIKE 'A%' OR nome LIKE 'I%'
```

E)

```
SELECT
    world
FROM nome
WHERE nome LIKE 'A%' OR nome LIKE 'I%'
```

O que faz a query abaixo?

```
SELECT  
    ID,  
    PROFISSAO,  
    NOME_EMPREGADO  
FROM EMPREGADOS  
WHERE PROFISSAO LIKE '%Engenheiro de pesquisa de desenvolvimento';
```

O que faz a query abaixo?

```
SELECT  
    ID,  
    PROFISSAO,  
    NOME_EMPREGADO  
FROM EMPREGADOS  
WHERE PROFISSAO LIKE '%Engenheiro de pesquisa de desenvolvimento';
```

Resposta:

Ela traz o id, a profissão e o nome do empregado da tabela EMPREGADOS, mas filtrando apenas aqueles cuja profissão possui ‘Engenheiro de pesquisa de desenvolvimento’ no desritivo da profissão’



Qual a diferença entre as duas queries abaixo?

```
SELECT  
    NOME_EMPREGADO,  
    SOBRENOME_EMPREGADO  
FROM EMPREGADOS  
WHERE SOBRENOME_EMPREGADO LIKE 'Ja%mes';
```

```
SELECT  
    NOME_EMPREGADO,  
    SOBRENOME_EMPREGADO  
FROM EMPREGADOS  
WHERE SOBRENOME_EMPREGADO LIKE 'Ja_mes';
```

Qual a diferença entre as duas queries abaixo?

```
SELECT
    NOME_EMPREGADO,
    SOBRENOME_EMPREGADO
FROM EMPREGADOS
WHERE SOBRENOME_EMPREGADO LIKE 'Ja%mes' ;
```

```
SELECT
    NOME_EMPREGADO,
    SOBRENOME_EMPREGADO
FROM EMPREGADOS
WHERE SOBRENOME_EMPREGADO LIKE 'Ja_mes' ;
```

Resposta:

Na primeira traz o nome e o sobrenome dos empregados da tabela EMPREGADOS cujo sobrenome possui 'Ja' + 'qualquer caractere, qualquer número de vezes' + 'mes', a segunda traz aqueles cujo sobrenome Possui 'Ja' + 'qualquer caractere, apenas uma vez' + 'mes'

Se tivermos na tabela os valores: [baseball, footbal, soccer, basketball, criket].

Qual será a resposta de: LIKE ‘_s%l’?

Se tivermos na tabela os valores: [baseball, footbal, soccer, basketball, criket].

Qual será a resposta de: LIKE ‘__s%l’?

Resposta:

‘Baseball’ e ‘basketball’

Escrever um padrão de busca que começa com “B” e termina com “ll”, por exemplo: Bill ou Ball

Escrever um padrão de busca que começa com “B” e termina com “ll”, por exemplo: Bill ou Ball

Resposta:

LIKE ‘B%ll’

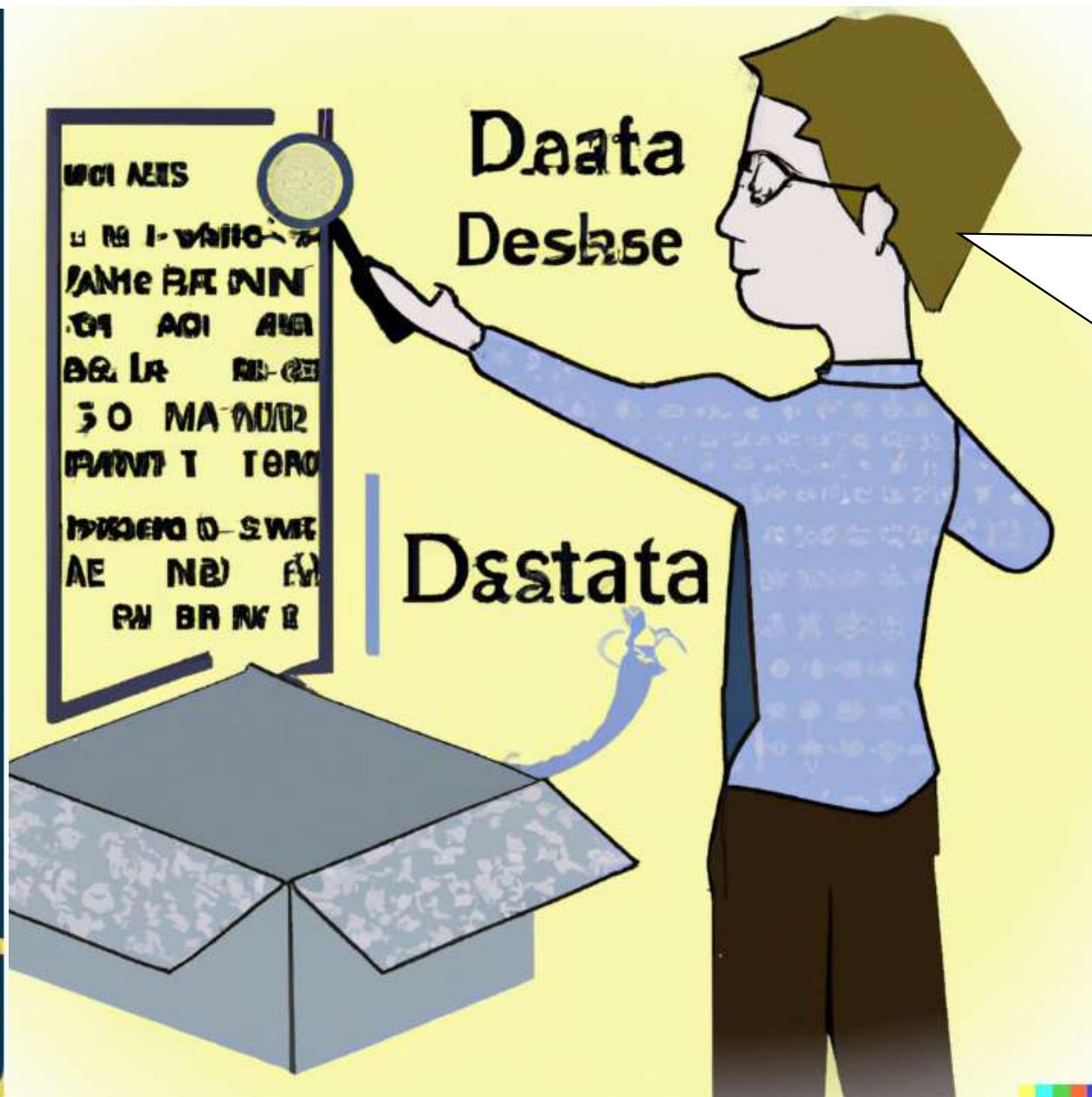
ORDER BY



A cláusula **ORDER BY** no SQL é usada para **classificar** o conjunto de resultados de uma instrução **SELECT** em **ordem crescente ou decrescente**.

Você pode usá-lo para classificar a tabela resposta por **uma ou mais colunas**.





Você deve usar **ORDER BY** sempre que quiser **classificar a tabela** resultado de uma instrução SELECT.

Isso pode ser útil para **organizar os dados** ou para facilitar a localização de algumas linhas na tabela resultado.

Aqui está um exemplo de uso do ORDER BY

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME  
FROM CLIENTES  
ORDER BY SOBRENOME ASC;
```

id	nome	sobrenome	salario
4	Lorena	Ferreira	2.400
3	Alan	Madureira	2.400
5	Nicole	Pereira	2.400
2	Laura	Silva	3.000
1	Alan	Williams	3.000

Neste exemplo a cláusula ORDER BY classifica o conjunto de resultados pela coluna sobrenome em ordem crescente.

A palavra-chave ASC especifica a ordem crescente e é a ordem padrão se você a omitir.

Você também pode usar a palavra-chave DESC para especificar a ordem decrescente.

Usando o ORDER BY

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME  
FROM CLIENTES  
ORDER BY SOBRENOME ASC;
```

ASC = ordem ascendente;
DESC = ordem descendente;



Então eu posso ordenar tanto na ordem crescente (ASC) como decrescente (DESC)

Usando o ORDER BY DESC

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME  
FROM CLIENTES  
ORDER BY SOBRENOME DESC;
```

id	nome	sobrenome	salario
1	Alan	Williams	3.000
2	Laura	Silva	3.000
5	Nicole	Pereira	2.400
3	Alan	Madureira	2.400
4	Lorena	Ferreira	2.400

Repare que neste caso a coluna sobrenome está ordenada do maior para o menor

Neste exemplo ordenamos por duas colunas

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME,  
    SALARIO  
FROM CLIENTES  
ORDER BY SALARIO ASC, NOME DESC;
```

The diagram illustrates the execution flow of a SQL query. On the left, a dark rectangular box contains the query code. An arrow points from this box to a central light gray box containing explanatory text. Another arrow points from this text box to the right, where a table is displayed. The table has four columns: id, nome, sobrenome, and salario. The rows are ordered by salary (ASC) and name (DESC). A blue arrow labeled '1' points upwards from the last row to the first row, indicating the primary sort order. A blue arrow labeled '2' points downwards from the first row to the last row, indicating the secondary sort order.

id	nome	sobrenome	salario
	Nicole	Pereira	2.400
	Lorena	Ferreira	2.400
	Alan	Madureira	2.400
	Laura	Silva	3.000
	Alan	Williams	3.000

Primeiro o salário do menor para o maior E DEPOIS o nome do maior para o menor



Em alguns casos pode ser útil saber ordenar por duas ou mais colunas...



Quando não existe o LIMIT, o ORDER BY **sempre** vai ser a última linha da query.

Depois do ORDER BY vem a coluna que se quer ordenar e o sentido, que pode ser ASC ou DESC.

O padrão, se não escrever nada, é ASC.

Listar todos os diretores (sem duplicatas) e ordenados por ordem alfabética

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Copolla	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Listar todos os diretores (sem duplicatas) e ordenados por ordem alfabética

```
SELECT
    DISTINCT DIRETOR
FROM FILMES
ORDER BY 1;
```

Listar os últimos 4 filmes ordenados do mais recente para o menos recente

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Listar os últimos 4 filmes ordenados do mais recente para o menos recente

```
SELECT
  *
FROM FILMES
ORDER BY ANO DESC
LIMIT 4;
```

Listar os primeiros 5 filmes da Pixar ordenados em ordem alfabética

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Resposta: Listar os primeiros 5 filmes ordenados em ordem alfabética

```
SELECT
  *
FROM FILMES
ORDER BY TITULO ASC
LIMIT 5;
```

Qual a diferença das duas queries abaixo?

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME  
FROM CLIENTES  
ORDER BY NOME, SOBRENOME;
```

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME  
FROM CLIENTES  
ORDER BY NOME DESC, SOBRENOME DESC;
```

Qual a diferença das duas queries abaixo?

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME  
FROM CLIENTES  
ORDER BY NOME, SOBRENOME;
```

```
SELECT  
    ID,  
    NOME,  
    SOBRENOME  
FROM CLIENTES  
ORDER BY NOME DESC, SOBRENOME DESC;
```

Resposta:

Na primeira trazemos o id, nome e sobrenome da tabela CLIENTES, ordenados pelo NOME e SOBRENOME em ordem crescente.

Na segunda trazemos as mesmas colunas da mesma tabela, mas ordenando o NOME e o SOBRENOME em ordem decrescente



NULOS E TRATAMENTO DE ERROS



Por que estudar
NULOS?

É o que aparece quando não
temos o valor, e isso acontece
com frequência.

Outro erro comum é que quando
uma coluna tem NULL ela pode
ser fonte de erros ao juntar duas
colunas.



Vale lembrar que **NULL** não é zero e
não é uma “string” vazia.

E também que no filtro **NÃO** é possível
fazer ‘=NULL’



Olá, meu nome é Marco e tenho uma mercearia.

Infelizmente a pessoa do cadastro não colocou preço em todos os produtos. Poderia identificar quais produtos são estes?

Da tabela a seguir como filtramos apenas as linhas nulas?

id	nome	preco
1	manteiga	NULL
2	leite	2,35
3	pão	3,25
4	queijo	NULL

Da tabela a seguir como filtramos apenas as linhas nulas?

```
SELECT
  *
FROM PRODUTO
WHERE PRECO IS NULL;
```

id	nome	preco
1	manteiga	NULL
4	queijo	NULL

Da tabela a seguir como filtramos apenas as linhas não nulas?

id	nome	preco
1	manteiga	NULL
2	leite	2,35
3	pão	3,25
4	queijo	NULL

Da tabela a seguir como filtramos apenas as linhas não nulas?

```
SELECT
  *
FROM PRODUTO
WHERE PRECO IS NOT NULL;
```

id	nome	preco
2	leite	2,35
3	pão	3,25

Na combinação de **colunas**, NULL com qualquer outro tipo de dado retorna NULL

Exemplos:

- $2 + \text{NULL}$ retorna NULL
- $2 * \text{NULL}$ retorna NULL
- 'Olá' || NULL retorna NULL
- NULL + interval '1 day' retorna NULL
- TRUE and NULL retorna NULL





Existem várias razões, mas se destacam dois motivos:

- 1- O dado não estava disponível na hora da inserção;**
- 2- Pode ser o resultado de um LEFT JOIN que não tinha valor na tabela da esquerda;**



NULL não necessariamente é um erro, pois na prática surgem muitos valores NULL.

Como não tem como escapar, podemos fazer tratamentos de valores NULL.

Olá! É você que vai nos ajudar a fazer um relatório de quem deve pagar as multas?



Estudo de Caso: Como tratar NULLs?

id_motorista	valor_multa	motivo	multas_nao_pagas
1	150	velocidade	NULL
2	500	farol vermelho	5040
3	150	velocidade	250
4	100	velocidade	0
5	100	uso celular	0

Nesta tabela de multas temos:

- o id do motorista
- o valor da multa
- a razão da multa
- o total de multas não pagas



Se quisermos fazer um tratamento de NULL e substitui-lo por outro valor, podemos usar a função COALESCE.



A função COALESCE no SQL analisa uma lista de valores e retorna o primeiro valor não nulo da lista. Você pode usá-lo para substituir um valor nulo por um valor padronizado.

Aqui está um exemplo de uso do COALESCE

```
SELECT  
    COALESCE(COLUNA1, 'Valor Padrão')  
FROM TABELA;
```

Neste caso, para todas as linhas da COLUNA1 que estiverem nulas, cada nulo receberá ‘Valor Padrão’.

Como fazer uma query para trazer o valor total de multas (não pagas + atual) por id de motorista?

Como fazer uma query para trazer o valor total de multas (não pagas + atual) por id de motorista?

```
SELECT
    ID_MOTORISTA,
    VALOR_MULTA,
    VALOR_MULTA + MULTAS_NAO_PAGAS AS VALOR_TOTAL
FROM MULTAS;
```

Se fizermos desta maneira, o resultado será:

id_motorista	valor_multa	valor_total
1	150	NULL
2	500	5540
3	150	400
4	100	100
5	100	100

Corrigindo pelo COALESCE, temos:

```
SELECT
    ID_MOTORISTA,
    VALOR_MULTA,
    VALOR_MULTA + COALESCE(MULTAS_NAO_PAGAS,0) AS VALOR_TOTAL
FROM MULTAS;
```

Se fizermos desta maneira, o resultado será:

id_motorista	valor_multa	valor_total
1	150	150
2	500	5540
3	150	400
4	100	100
5	100	100

Além das multas não pagas temos o nível de infração a ser considerado

id_motorista	valor_multa	motivo	nivel_infracao	multas_nao_pagas
1	150	velocidade	2	NULL
2	500	farol vermelho	1	5040
3	150	velocidade	2	250
4	100	velocidade	2	0
5	100	uso celular	NULL	0

Como fazer uma query que traga a tabela resultado abaixo?

Partir desta:

id_motorista	valor_multa	motivo	nivel_infracao	multas_nao_pagas
1	150	velocidade	2	NULL
2	500	farol vermelho	1	5040
3	150	velocidade	2	250
4	100	velocidade	2	0
5	100	uso celular	NULL	0

Chegar nesta:

id_motorista	valor_multa	valor_total	valor_30d
1	150	150	300
2	500	5.540	5.540
3	150	400	550
4	100	100	200
5	100	100	100

Valor da multa

valor_multa + multas não pagas

valor_multa * Nível
Infração + multas não pagas

Resposta

```
SELECT
    ID_MOTORISTA,
    VALOR_MULTA,
    VALOR_MULTA + COALESCE(MULTAS_NAO_PAGAS,0) AS VALOR_TOTAL,
    VALOR_MULTA*COALESCE(NIVEL_INFRACAO,1)+ COALESCE(MULTAS_NAO_PAGAS,0) AS VALOR_30D
FROM MULTAS;
```

id_motorista	valor_multa	valor_total	valor_30d
1	150	150	300
2	500	5.540	5.540
3	150	400	550
4	100	100	200
5	100	100	100

O que faz as queries abaixo?

```
SELECT *  
FROM clientes  
WHERE email IS NULL;
```

```
SELECT *  
FROM funcionarios  
WHERE data_contratacao IS NULL;
```

O que faz as queries abaixo?

```
SELECT *  
FROM clientes  
WHERE email IS NULL;
```

```
SELECT *  
FROM funcionarios  
WHERE data_contratacao IS NULL;
```

Resposta:

Na primeira trazemos todas as colunas da tabela CLIENTES mas filtrando apenas aquelas linhas cujo e-mail é nulo

Na segunda todas as colunas da tabela FUNCIONARIOS mas filtrando apenas aquelas linhas cuja data de contratação é nula

Suponha que tenha uma tabela chamada produtos com uma coluna descrição, como trazer as os produtos que não tem descrição?

Suponha que tenha uma tabela chamada produtos com uma coluna descrição, como trazer as os produtos que não tem descrição?

Resposta:

```
SELECT *  
FROM produtos  
WHERE descricao IS NULL;
```

Qual a diferença das duas queries abaixo?

```
SELECT  
    *  
FROM LISTA_DE_ITENS  
WHERE NOME_ITEM IS NULL;
```

```
SELECT  
    *  
FROM LISTA_DE_ITENS  
WHERE NOME_ITEM IS NOT NULL;
```

Qual a diferença das duas queries abaixo?

```
SELECT  
    *  
FROM LISTA_DE_ITENS  
WHERE NOME_ITEM IS NULL;
```

```
SELECT  
    *  
FROM LISTA_DE_ITENS  
WHERE NOME_ITEM IS NOT NULL;
```

Na primeira trazemos todas as colunas da tabela LISTA DE ITENS mas filtrando apenas aquelas cujo nome do item é nulo

Na segunda trazemos todas as colunas da tabela LISTA DE ITENS mas filtrando apenas aquelas cujo nome do item NÃO é nulo

Por que a query abaixo está errada?

```
SELECT
  *
FROM LISTA_DE_ITENS
WHERE NOME_ITEM = NULL;
```

Por que a query abaixo está errada?

```
SELECT
  *
FROM LISTA_DE_ITENS
WHERE NOME_ITEM = NULL;
```

Resposta:

O filtro correto para trazer nulos é IS NULL, e não '= NULL'

O que faz a query abaixo?

```
SELECT
  *
FROM PRODUTOS
WHERE COR IS NULL OR COR <> 'Azul';
```

O que faz a query abaixo?

```
SELECT
  *
FROM PRODUTOS
WHERE COR IS NULL OR COR <> 'Azul';
```

Resposta:

Traz todas as colunas da tabela PRODUTOS cuja cor é nula ou não é azul

Quando usar o COALESCE?

Resposta:

Quando usar o COALESCE?

Resposta:

Quando queremos tratar o valor nulo, substituindo-o por algum outro valor

Traga o nome e a data de entrega da tabela “pedidos”, mas exiba “Pendente” para pedidos que ainda não foram entregues:

Traga o nome e a data de entrega de todos os pedidos, mas exiba "Pendente" para pedidos que ainda não foram entregues:

Resposta:

```
SELECT
    nome,
    COALESCE(data_entrega , 'Pendente') AS data_entrega_formatada
FROM pedidos;
```

Selecione o nome e o preço da tabela “produtos”, mas exiba “Sem preço” para produtos que não possuem um preço definido

Selecione o nome e o preço de todos os produtos, mas exiba "Sem preço" para produtos que não possuem um preço definido

Resposta:

```
SELECT
    nome,
    COALESCE(preco, 'Sem preço') AS preco_formatado
FROM produtos;
```

Qual a resposta da expressão: NULL > NULL?

Resposta:

Qual a resposta da expressão: NULL > NULL?

Resposta:

A resposta é NULL

Qual a resposta da expressão: NULL > NULL
AND TRUE?

Resposta:

Qual a resposta da expressão: NULL > NULL
AND TRUE?

Resposta:

A resposta é NULL

Qual a resposta da expressão: NULL > NULL
OR TRUE?

Resposta:

Qual a resposta da expressão: NULL > NULL
OR TRUE?

Resposta:

A resposta é TRUE

Desafio: Colaboradores e projetos

As tabelas ao lado são referentes à empresa de projetos em que você trabalha.

A sequência de perguntas a seguir ajudarão seu chefe a entender melhor a situação na qual a empresa se encontra.

Tabela funcionarios

id_empregado	primeiro_nome	sobrenome	departamento	salario
1001	John	Smith	HR	50.000
1002	Emma	Johnson	Sales	60.000
1003	Michael	Brown	IT	55.000
1004	Olivia	Davis	HR	52.000
1005	Sophia	Garcia	Accounting	48.000

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Listar os primeiros e últimos nomes de todos os funcionários

Query ou consulta

Tabela funcionarios

id_empregado	primeiro_nome	sobrenome	departamento	salario
1001	John	Smith	HR	50.000
1002	Emma	Johnson	Sales	60.000
1003	Michael	Brown	IT	55.000
1004	Olivia	Davis	HR	52.000
1005	Sophia	Garcia	Accounting	48.000

Tabela Resposta

primeiro_nome	sobrenome
John	Smith
Emma	Johnson
Michael	Brown
Olivia	Davis
Sophia	Garcia

Resposta: Listar os primeiros e últimos nomes de todos os funcionários

```
SELECT  
    primeiro_nome,  
    sobrenome  
FROM funcionarios;
```

Tabela funcionarios

id_empregado	primeiro_nome	sobrenome	departamento	salario
1001	John	Smith	HR	50.000
1002	Emma	Johnson	Sales	60.000
1003	Michael	Brown	IT	55.000
1004	Olivia	Davis	HR	52.000
1005	Sophia	Garcia	Accounting	48.000

Tabela Resposta

primeiro_nome	sobrenome
John	Smith
Emma	Johnson
Michael	Brown
Olivia	Davis
Sophia	Garcia

Listar todas as colunas dos projetos com receita maior que 40.000

Query ou consulta

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Resposta: Listar todas as colunas dos projetos com receita maior que 40.000

```
SELECT
  *
FROM projetos
WHERE receita > 40000;
```

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Listar os nomes de departamentos dos projetos com receita entre 100.000 e 150.000.

Query ou consulta

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

nome_departamento
IT

Resposta: Listar os nomes de departamentos dos projetos com receita entre 100.000 e 150.000.

```
SELECT
    nome_departamento
FROM projetos
WHERE receita BETWEEN 100000 AND 150000;
```

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

nome_departamento
IT

Listar os IDs de projeto para os projetos que começaram em ou antes de 1º de julho de 2024.

Query ou consulta

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

id_projeto
P1
P2
P3
P4
ADT4MFIA

Resposta: Listar os IDs de projeto para os projetos que começaram em ou antes de 1º de julho de 2004.

```
SELECT
    id_projeto
FROM projetos
WHERE data_inicio <= '2024-07-01';
```

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

id_projeto
P1
P2
P3
P4
ADT4MFIA

Listar os IDs de projeto para os projetos que não tem data_fim
(ou seja, estão em andamento)

Query ou consulta

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

id_projeto
P4

Resposta: Listar os IDs de projeto para os projetos que não tem data_fim (ou seja, estão em andamento)

```
SELECT
    id_projeto
FROM projetos
WHERE data_fim IS NULL;
```

Tabela projetos

id_projeto	nome_projeto	receita	data_inicio	data_fim	nome_departamento
P1	Project A	45.000	10/01/2022	15/06/2022	RH
P2	Project B	80.000	20/03/2022	30/09/2022	Vendas
P3	Project C	120.000	05/05/2022	31/12/2022	TI
P4	Project D	30.000	30/06/2022	NULL	RH
ADT4MFIA	Project X	75.000	15/02/2023	31/10/2023	Vendas

Tabela Resposta

id_projeto
P4

Listar todas as informações sobre funcionários com sobrenomes que terminam com 'son'.

Query ou consulta

Tabela funcionarios

id_empregado	primeiro_nome	sobrenome	departamento	salario
1001	John	Smith	HR	50.000
1002	Emma	Johnson	Sales	60.000
1003	Michael	Brown	IT	55.000
1004	Olivia	Davis	HR	52.000
1005	Sophia	Garcia	Accounting	48.000

Tabela Resposta

id_empregado	primeiro_nome	sobrenome	departamento	salario
1002	Emma	Johnson	Sales	60.000

Resposta: Listar todas as informações sobre funcionários com sobrenomes que terminam com 'son'.

```
SELECT
  *
FROM funcionarios
WHERE sobrenome LIKE '%son';
```

Tabela funcionarios

id_empregado	primeiro_nome	sobrenome	departamento	salario
1001	John	Smith	HR	50.000
1002	Emma	Johnson	Sales	60.000
1003	Michael	Brown	IT	55.000
1004	Olivia	Davis	HR	52.000
1005	Sophia	Garcia	Accounting	48.000

Tabela Resposta

id_empregado	primeiro_nome	sobrenome	departamento	salario
1002	Emma	Johnson	Sales	60.000

Listar o ID e sobrenome de todos os funcionários que trabalham para o departamento Accounting e ganham menos de 50.000.

Query ou consulta

Tabela funcionarios

id_empregado	primeiro_nome	sobrenome	departamento	salario
1001	John	Smith	HR	50.000
1002	Emma	Johnson	Sales	60.000
1003	Michael	Brown	IT	55.000
1004	Olivia	Davis	HR	52.000
1005	Sophia	Garcia	Accounting	48.000

Tabela Resposta

id_empregado	sobrenome
1005	Garcia

Resposta: Listar o ID e sobrenome de todos os funcionários que trabalham para o departamento Accounting e ganham menos de 50.000.

```
SELECT
    id_empregado,
    sobrenome
FROM funcionarios
WHERE departamento = 'Accounting'
AND salario < 50000;
```

Tabela funcionarios

id_empregado	primeiro_nome	sobrenome	departamento	salario
1001	John	Smith	HR	50.000
1002	Emma	Johnson	Sales	60.000
1003	Michael	Brown	IT	55.000
1004	Olivia	Davis	HR	52.000
1005	Sophia	Garcia	Accounting	48.000

Tabela Resposta

id_empregado	sobrenome
1005	Garcia

Desafio: Como escrever uma query para encontrar os testes que foram concluídos? Se a data de conclusão for nula (comp_date), significa que o teste ainda não foi concluído.

testresults

<code>test_name</code>	<code>test_step</code>	<code>comp_date</code>
Reading Skills	1	2021-01-01
Reading Skills	2	2021-01-02
Reading Skills	3	2021-01-03
Reading Skills	4	2021-01-04
Reading Skills	5	2021-01-05
Math Skills	1	2021-01-06
Math Skills	2	2021-01-07
Math Skills	3	2021-01-08
Math Skills	4	2021-01-09
Math Skills	5	2021-01-10
Math Skills	6	2021-01-11

Resposta Desafio

```
SELECT DISTINCT test_name  
FROM TestResults  
WHERE comp_date IS NOT NULL;
```

FUNÇÕES DE NÚMERO

Introdução às funções de NÚMERO

Existem várias funções numéricas em SQL, cada uma com sua própria finalidade.

Aqui estão alguns exemplos:

- **ROUND**: Arredonda um número para o número especificado de casas decimais.
- **FLOOR**: Arredonda um número para baixo até o inteiro mais próximo.
- **ABS**: Retorna o valor absoluto de um número, ou seja, o número sem sinal (positivo ou negativo).

Essas funções podem ser úteis para realizar cálculos matemáticos em SQL, seja em consultas simples ou em relatórios mais complexos.

Função de Número: ROUND

A função ROUND é usada para arredondar um número para um determinado número de casas decimais. O arredondamento é baseado no dígito à direita do último dígito desejado. Se o dígito à direita for **maior ou igual a 5**, o número será arredondado **para cima, caso contrário, será arredondado para baixo**.

```
SELECT
    id,
    valor,
    ROUND(valor, 1) AS valor_arredondado_com_1_casa_decimal
FROM numeros;
```

id	valor	valor_arredondado_com_1_casa_decimal
1	3.14	3.1
2	5.7	5.7
3	-2.8	-2.8
4	10.1	10.1
5	-7.9	-7.9
6	0.0	0.0
7	2.718	2.7
8	9.99	10.0
9	6.5	6.5
10	-4.2	-4.2

Função de Número: FLOOR

A função FLOOR é utilizada para **arredondar um número para baixo**, ou seja, retornar o maior número inteiro menor ou igual ao número informado como argumento. Por exemplo, o número 3.8 seria arredondado para 3.

```
SELECT
    id,
    valor,
    FLOOR(valor) AS valor_arredondado_para_baixo
FROM numeros;
```

id	valor	valor_arredondado_para_baixo
1	3.14	3.0
2	5.7	5.0
3	-2.8	-3.0
4	10.1	10.0
5	-7.9	-8.0
6	0.0	0.0
7	2.718	2.0
8	9.99	9.0
9	6.5	6.0
10	-4.2	-5.0

Função de Número: ABS

A função ABS retorna o **valor absoluto de um número**, ou seja, o valor **sem o sinal negativo**, se houver. Por exemplo, o valor absoluto de -5 é 5, e o valor absoluto de 5 é 5.

```
SELECT
    id,
    valor,
    ABS(valor) AS valor_absoluto
FROM numeros;
```

<code>id</code>	<code>valor</code>	<code>valor_absoluto</code>
1	3.14	3.14
2	-5.67	5.67
3	2.35	2.35
4	-4.89	4.89
5	6.78	6.78
6	-9.01	9.01
7	1.23	1.23
8	-7.56	7.56
9	4.09	4.09
10	-2.77	2.77

Qual a resposta da expressão abaixo?

SELECT 5*8+2/ ABS(NULL)

Qual a resposta da expressão abaixo?

```
SELECT 5*8+2/ ABS(NULL)
```

Resposta:

O resultado da expressão seria NULL.

Quando um cálculo envolve um valor NULL, o resultado também será NULL.

Como você faria uma query para avaliar que a expressão $e(\ln x) = x$ é sempre verdadeira?

Como você faria uma query para avaliar que a expressão $e(\ln x) = x$ é sempre verdadeira?

Resposta:

Para testar as funções SQL $e(x)$ e $\ln(x)$, você pode escrever a seguinte consulta:

`SELECT eln(x), x FROM sua_tabela;`

Substitua "sua_tabela" pelo nome da tabela que contém a coluna "x".

Essa consulta retornará os valores da coluna "x"

e seus respectivos valores $\ln(x)$, que é o logaritmo natural de x.

FUNÇÕES DE TEXTO

Introdução às funções de TEXTO

- Assim como em qualquer outra linguagem, SQL também tem funções especiais de texto;
 - LOWER
 - UPPER
 - SUBSTR
 - REPLACE
 - TRIM
 - CONCAT
- Vale ressaltar que o único momento em que SQL é caso-sensitivo é dentro das aspas, ou seja, na comparação de strings;

Função de Texto: UPPER/LOWER

A função UPPER converte todos os caracteres de uma string em letras maiúsculas.
Já a função LOWER converte todos os caracteres de uma string em letras minúsculas.

```
SELECT
    id,
    upper(nome_completo) as nome_em_maiusculo,
    lower(nome_completo) as nome_em_minusculo,
    idade
FROM pessoas;
```

	id	nome_em_maiusculo	nome_em_minusculo	idade
1	JOÃO DA SILVA	joão da silva	30	
2	MARIA LIMA	maria lima	25	
3	PEDRO SOUZA	pedro souza	42	
4	ANA SANTOS	ana santos	37	
5	RAFAEL SILVEIRA	rafael silveira	28	
6	JULIA GOMES	julia gomes	21	
7	MARCOS OLIVEIRA	marcos oliveira	35	
8	CAMILA SOUSA	camila sousa	29	
9	LUIS COSTA	luis costa	46	
10	FERNANDA RAMOS	fernanda ramos	31	

Função de Texto: SUBSTR

SUBSTR é uma das funções mais importantes do SQL;

Ela faz o mesmo que o EXT.TEXTO do Excel e retorna uma string subconjunto da string original, definindo a coluna, a posição inicial e a quantidade de strings a partir da posição dada;

```
SELECT
    id,
    substr(nome_completo, 1, 3) as tres_primeiros_caracteres,
    idade
FROM pessoas;
```

	<code>id</code>	<code>tres_primeiros_caracteres</code>	<code>idade</code>
1	1	Joã	30
2	2	Mar	25
3	3	Ped	42
4	4	Ana	37
5	5	Raf	28
6	6	Jul	21
7	7	Mar	35
8	8	Cam	29
9	9	Lui	46
10	10	Fer	31

Função de Texto: REPLACE

A função REPLACE substitui uma sequência de caracteres por outra dentro de uma string. Ela recebe três argumentos: a string original, a sequência de caracteres a ser substituída e a nova sequência de caracteres a ser usada como substituta.

```
SELECT
    id,
    REPLACE(REPLACE(primeiro_nome, 'a', ''), 'e', '') as primeiro_nome_sem_a_e,
    sobrenome,
    idade
FROM pessoas;
```

	id	primeiro_nome_sem_a_e	sobrenome	idade
1	1	João	Silva	30
2	2	Mir	Lima	25
3	3	Pdro	Souza	42
4	4	An	Silva	37
5	5	Rf	Silva	28

Função de Texto: TRIM

A função TRIM remove espaços em branco ou outros caracteres especificados do início e/ou fim de uma string. Ela pode receber até dois argumentos: a string original e os caracteres a serem removidos.

id	primeiro_nome	sobrenome	idade
1	João	Silva	30
2	Maria	Lima	25
3	Pedro	Souza	42
4	Ana	Santos	37
5	Rafael	Silveira	28
6	Luiz	Kim	33
7	Gabriela	Almeida	24
8	Marcos	Leite	39
9	Laura	Pereira	27
10	Bruno	Mendonça	36

```
SELECT
    id,
    trim(primeiro_nome) as primeiro_nome,
    trim(sobrenome) as sobrenome,
    idade
FROM pessoas;
```

id	primeiro_nome	sobrenome	idade
1	João	Silva	30
2	Maria	Lima	25
3	Pedro	Souza	42
4	Ana	Santos	37
5	Rafael	Silveira	28
6	Luiz	Kim	33
7	Gabriela	Almeida	24
8	Marcos	Leite	39
9	Laura	Pereira	27
10	Bruno	Mendonça	36

Função de Texto: CONCAT

A função CONCAT concatena duas ou mais strings em uma única string. Ela recebe dois ou mais argumentos, que representam as strings que devem ser concatenadas, e retorna uma nova string que é a concatenação das strings de entrada.

id	primeiro_nome	sobrenome	idade
<hr/>			
1	João	Silva	30
2	Maria	Lima	25
3	Pedro	Souza	42
4	Ana	Santos	37
5	Rafael	Silveira	28

```
SELECT
    id,
    concat(primeiro_nome, ' ', sobrenome) as nome_completo,
    idade
FROM pessoas;
```

id	nome_completo	idade
<hr/>		
1	João Silva	30
2	Maria Lima	25
3	Pedro Souza	42
4	Ana Santos	37
5	Rafael Silveira	28

Escreva uma query para trazer a primeira letra de cada palavra da coluna seu_campo?

```
SELECT SUBSTRING(seu_campo, 1, 1) AS primeira_letra FROM  
sua_tabela;
```

Escreva uma query para trazer a primeira palavra de cada linha da coluna seu_campo?

```
SELECT SUBSTRING(seu_campo, 1, POSITION(' ' IN seu_campo) - 1) AS  
primeira_palavra FROM sua_tabela;
```

Desafio: Insights sobre projetos

1. Listar todos os nomes de funcionários em um campo chamado "name"
2. Listar todos os códigos de departamento atribuídos a um projeto, removendo duplicatas
3. Precisamos criar uma lista de nomes abreviados de projetos. Cada nome abreviado concatenará os primeiros três caracteres da descrição do projeto, um hífen e o código do departamento. Todos os caracteres devem estar em maiúsculas (por exemplo, EMP-ADMIN).
4. Para cada projeto, listar o ID e o ano em que o projeto foi iniciado. Ordenar os resultados em ordem crescente por ano.
5. Se cada funcionário receber um aumento de 5%, encontrar o sobrenome e o novo salário dos funcionários que ganharão mais de \$50.000.
6. Para todos os funcionários no departamento HDWRE, listar seu ID, primeiro nome, sobrenome e salário após um aumento de 10%. A coluna de salário no resultado deve ser denominada "Próximo Ano".

Tabela employees

employee_id	first_name	last_name	department_code	salary
1	John	Doe	ACCNT	50.000
2	Jane	Smith	CNSLT	60.000
3	Mike	Johnson	HDWRE	55.000
4	Lisa	Brown	ACTNG	45.000
5	Mark	Lee	CNSLT	70.000
6	Sarah	Williams	HDWRE	58.000
7	David	Miller	ACTNG	48.000
8	Emily	Jones	CNSLT	65.000
9	Michael	Wilson	HDWRE	60.000
10	Jennifer	Taylor	ACCNT	52.000

Tabela projects

project_id	description	start_date	end_date	revenue	department_code
1	Marketing Campaign	10/01/2022	15/03/2022	80.000	CNSLT
2	Product Development	20/02/2022	30/06/2022	120.000	HDWRE
3	Financial Analysis	05/03/2022	30/04/2022	60.000	ACCNT
4	Sales Strategy	15/04/2022	NULL	50.000	ACTNG
5	IT Infrastructure	01/05/2022	31/08/2022	90.000	CNSLT
6	Customer Support	10/06/2022	15/09/2022	70.000	HDWRE
7	HR Recruitment	20/07/2022	30/11/2022	100.000	CNSLT
8	Inventory Management	01/08/2022	NULL	75.000	HDWRE
9	Market Research	15/09/2022	31/12/2022	85.000	CNSLT
10	Financial Reporting	10/10/2022	28/02/2023	110.000	ACCNT

Resposta Desafio: Insights sobre projetos

```
SELECT CONCAT(first_name, ' ', last_name) AS name  
FROM employees;
```

Tabela Resposta

name
John Doe
Jane Smith
Mike Johnson
Lisa Brown
Mark Lee
Sarah Williams
David Miller
Emily Jones
Michael Wilsor

Resposta Desafio: Insights sobre projetos

```
SELECT DISTINCT department_code  
FROM projects;
```

Tabela Resposta

department_code
CNSLT
HDWRE
ACCNT
ACTNG

Resposta Desafio: Insights sobre projetos

```
SELECT CONCAT(UPPER(SUBSTRING(description, 1, 3)), '-', department_code)
| AS abbreviated_name
FROM projects;
```

Tabela Resposta

abbreviated_name
MAR-CNSLT
PRO-HDWRE
FIN-ACCT
SAL-ACTNG
IT -CNSLT
CUS-HDWRE
HR -CNSLT
INV-HDWRE
MAR-CNSLT
FIN-ACCT

Resposta Desafio: Insights sobre projetos

```
SELECT project_id, YEAR(start_date) AS project_year  
FROM projects  
ORDER BY project_year ASC;
```

Tabela Resposta

project_id	project_year
1	2022
2	2022
3	2022
4	2022
5	2022
6	2022
7	2022
8	2022
9	2022
10	2022

Resposta Desafio: Insights sobre projetos

```
SELECT last_name, salary * 1.05 AS new_salary  
FROM employees  
WHERE salary * 1.05 > 50000;
```

Tabela Resposta

last_name	new_salary
Smith	63.000
Johnson	57.750
Lee	73.500
Williams	60.900
Jones	68.250
Wilson	63.000

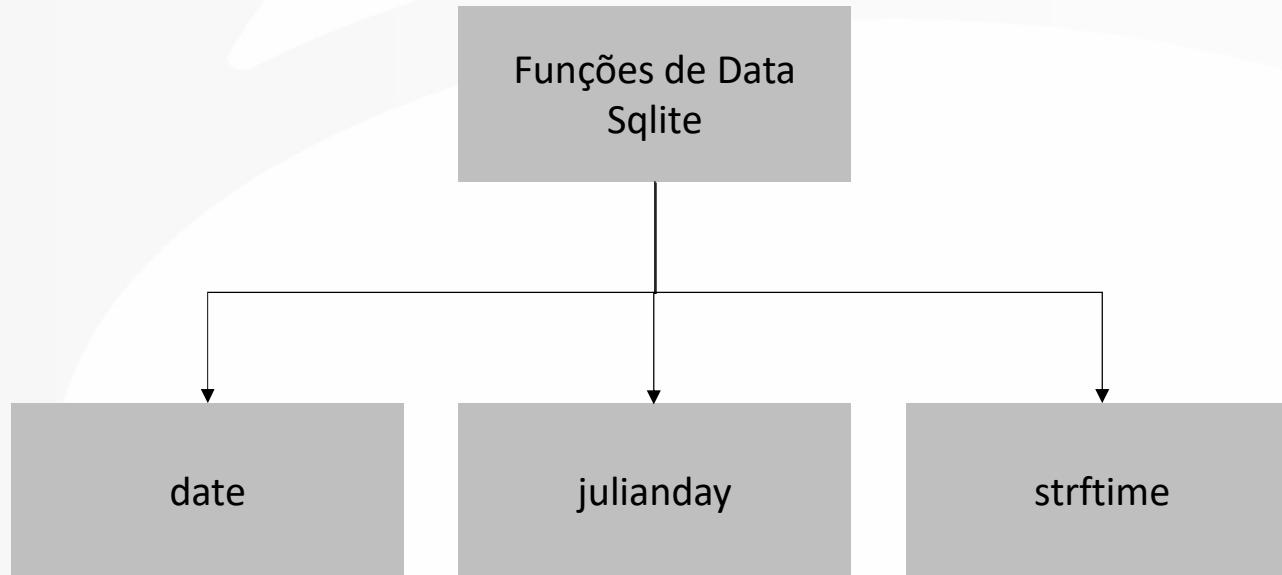
Resposta Desafio: Insights sobre projetos

```
SELECT employee_id, first_name, last_name, salary * 1.10 AS "Próximo Ano"  
FROM employees  
WHERE department_code = 'HDWRE';
```

Tabela Resposta

employee_id	first_name	last_name	salary	Próximo Ano
3	Mike	Johnson	55.000	60.500
6	Sarah	Williams	58.000	63.800
9	Michael	Wilson	60.000	66.000

FUNÇÕES DE DATA



Data mais um dia,
Mais um mês, etc

Diferença de datas
em dias

Extrai o ano, mês,
dia, hora, minuto,
segundo, etc...

Função de Data: DATE

A função **DATE + 1DAY** é usada para **adicionar um dia** a uma data específica. Em outras palavras, ela retorna a data resultante de adicionar um dia à data original. Por exemplo, se a data original for "2023-04-10", a função retornará "2023-04-11".

```
SELECT
    id,
    primeiro_nome,
    sobrenome,
    data_nascimento,
    date(data_nascimento, '+1 day') AS data_nascimento_mais_um_dia
FROM pessoas;
```

	<code>id</code>	<code>primeiro_nome</code>	<code>sobrenome</code>	<code>data_nascimento</code>	<code>data_nascimento_mais_um_dia</code>
1	1	João	Silva	1990-05-10	1990-05-11
2	2	Maria	Santos	1985-08-20	1985-08-21
3	3	Pedro	Souza	1998-11-30	1998-12-01
4	4	Ana	Oliveira	2002-02-15	2002-02-16
5	5	Lucas	Costa	1995-07-18	1995-07-19
6	6	Juliana	Pereira	1978-04-05	1978-04-06
7	7	Rafael	Carvalho	1989-12-23	1989-12-24
8	8	Fernanda	Rodrigues	1993-09-01	1993-09-02
9	9	Marcos	Almeida	2001-06-29	2001-06-30
10	10	Bruna	Lima	1999-03-14	1999-03-15

Outras formas de usar a função DATE

- "+2 days": Adiciona dois dias à data.
- "-1 month": Subtrai um mês da data.
- "+1 year": Adiciona um ano à data.
- "start of month": Retorna o primeiro dia do mês da data.
- "weekday 3": Retorna o primeiro dia da semana (segunda-feira) após a data.

Função de Data: JULIANDAY

A função julianday é usada para obter o número de dias desde o dia 01/01/4713 AC (ou 4713 BCE) até a data especificada. Isso permite a comparação de datas em formatos diferentes e também facilita cálculos de intervalos de tempo entre datas.

```
SELECT
    id,
    primeiro_nome,
    sobrenome,
    data_nascimento,
    julianday(data_nascimento) AS dias_julianos_nascimento
FROM pessoas;
```

id	primeiro_nome	sobrenome	data_nascimento	julian_day
1	João	Silva	1990-01-01	2447892.5
2	Maria	Santos	1985-05-15	2446183.5
3	Ana	Souza	1992-11-30	2448989.5
4	Pedro	Alves	1978-07-20	2443698.5
5	Julia	Costa	2000-03-05	2451574.5
6	Felipe	Rodrigues	1982-09-10	2445183.5
7	Sofia	Oliveira	1995-12-25	2450524.5
8	Lucas	Martins	1975-06-12	2442508.5
9	Mariana	Pereira	1998-08-01	2450655.5
10	Carlos	Carvalho	1991-04-18	2448599.5

Função de Data: STRFTIME

A função strftime é utilizada para **formatar uma data ou hora** em uma string com o formato desejado. Ela recebe dois argumentos: o primeiro é a string de formato e o segundo é a data ou hora que se deseja formatar. Por exemplo, a string de formato "%Y-%m-%d" formata a data no formato "ano-mês-dia".

```
SELECT
    id,
    primeiro_nome,
    sobrenome,
    data_nascimento,
    strftime('%Y', data_nascimento) AS ano_nascimento
FROM pessoas;
```

	id	primeiro_nome	sobrenome	data_nascimento	ano_nascimento
1	1	João	Silva	1990-05-15	1990
2	2	Maria	Santos	1985-12-31	1985
3	3	José	Oliveira	1995-08-22	1995

Formatos Suportados pela função STRFTIME

- %Y: ano com 4 dígitos
- %m: mês com 2 dígitos
- %d: dia com 2 dígitos
- %H: hora com 2 dígitos (formato de 24 horas)
- %M: minuto com 2 dígitos
- %S: segundo com 2 dígitos
- %j: dia do ano com 3 dígitos (entre 001 e 366)
- %w: dia da semana com 1 dígito (entre 0 e 6, sendo 0 o domingo)

Escreva uma query para trazer o dia de amanhã

Escreva uma query para trazer o dia de amanhã

Resposta:

```
SELECT DATE_ADD(CURRENT_DATE(), INTERVAL 1 DAY) AS  
DIA_DE_AMANHA;
```

Desafio: Insights sobre projetos

1. Encontrar o ID do projeto e a duração em dias entre as datas de início e término
2. Encontrar o ID do projeto e a receita média por dia para projetos com data de término não nula
3. Encontrar o ano de início de cada projeto (remover duplicatas)

Tabela projects

project_id	project_name	start_date	end_date	revenue
1	Robotic Spouse	15/01/2023	31/03/2023	25.000
2	Download Client	28/02/2023	15/05/2023	32.000
3	Mobile App	10/03/2023	NULL	15.000
4	Data Analytics	05/04/2023	NULL	18.000
5	AI Chatbot	01/05/2023	20/07/2023	30.000
6	Automation Tool	10/06/2023	25/08/2023	28.000
7	CRM Integration	15/07/2023	NULL	22.000
8	Sales Dashboard	05/08/2023	30/10/2023	40.000
9	E-commerce Redes	20/09/2023	10/12/2023	35.000
10	Inventory System	01/10/2023	NULL	27.000

Resposta Desafio: Insights sobre as vendas

```
SELECT
    project_id,
    CAST(JulianDay(end_date) - JulianDay(start_date)) AS INTEGER) AS duration
FROM projects;
```

Tabela Resposta

project_id	duration
1	30
2	60
3	45
4	90
5	120
6	60
7	150
8	60
9	45
10	180

Resposta Desafio: Insights sobre as vendas

```
SELECT
    project_id,
    revenue / CAST((JulianDay(end_date) - JulianDay(start_date)) AS INTEGER)
        AS avg_revenue_per_day
FROM projects
WHERE end_date IS NOT NULL;
```

Tabela Resposta

project_id	avg_revenue_per_day
1	500
2	1.500
3	
4	1.000
5	2.500
6	2.000
7	
8	1.500
9	
10	2.500

Resposta Desafio: Insights sobre as vendas

```
SELECT
    DISTINCT strftime('%Y', start_date) AS start_year
FROM projects;
```

Tabela Resposta

start_year

2023

CASE WHEN





Por que estudar o CASE WHEN?

Muitas vezes queremos categorizar uma variável.

Algo como dizer se aquele dado cai na classificação 'A', 'B' ou 'C'.

Ele também é uma alternativa ao COALESCE para tratar nulos

A ideia do CASE WHEN é selecionar uma condição que precisamos para reclassificá-la numa nova nomenclatura.

Por exemplo:

*Case when banana = 'amarela' then
'madura' else 'verde' end*

A leitura da query ficaria: para quando a coluna banana for igual à informação 'amarela', então classificamos como 'madura', caso contrário classificamos como 'verde'.

O END é inserido para definir o término daquela condição.



Sintaxe do CASE

```
CASE
    WHEN CONDICA01 THEN RESULTADO1
    WHEN CONDICA02 THEN RESULTADO2
    WHEN CONDICA03 THEN RESULTADO3
    ...
    ELSE RESULTADO_ELSE
END
```

Começamos com CASE

Seguido por WHEN logo depois vem a expressão condicional

Seguido por THEN. Depois do THEN vem o valor que a coluna vai receber quando aquela condição for verdadeira

Repetimos WHEN e THEN para cada caso

Se nenhum caso for satisfeito virá o valor depois do ELSE

CASE termina com END;



Aqui no nosso zoológico gerenciamos
todos os animais num banco de dados

E eu administro tudo!

Da tabela a seguir, como reclassificamos a informação da coluna ZOO, para quando for igual a 1 constar ‘Zoológico de SP’ e igual a 2 constar ‘Simba Safari’?

id	nome	zoo
1	Leao	1
2	Tigre	2
3	Koala	2
4	Girafa	1
5	Elefante	2

```
SELECT
  *,
  CASE
    WHEN ZOO = 1 THEN 'Zoológico de SP'
    WHEN ZOO = 2 THEN 'Simba Safári'
    ELSE 'Desconhecido'
  END
FROM ANIMAIS;
```

Note que podemos adicionar várias condições dentro de um mesmo CASE:

A leitura da query ficaria assim:

Para quando a coluna ZOO for igual a 1, então reclassifica como 'Zoológico de SP'.

Para quando a coluna ZOO for igual a 2, então reclassifica como 'Simba Safári'.

Caso contrário classifica como 'Desconhecido'.



```
SELECT
  *,
  CASE
    WHEN ZOO = 1 THEN 'Zoológico de SP'
    WHEN ZOO = 2 THEN 'Simba Safári'
    ELSE 'Desconhecido'
  END
FROM ANIMAIS;
```

id	nome	zoo	case
1	Leao	1	Zoológico de SP
2	Tigre	2	Simba Safári
3	Koala	2	Simba Safári
4	Girafa	1	Zoológico de SP
5	Elefante	2	Simba Safári

Ao final do CASE podemos colocar AS para dar um nome à coluna que acabamos de criar

```
SELECT
  *,
  CASE
    WHEN ZOO = 1 THEN 'Zoológico de SP'
    WHEN ZOO = 2 THEN 'Simba Safári'
    ELSE 'Desconhecido'
  END AS ZOOLOGICO
FROM ANIMAIS;
```

id	nome	zoo	ZOOLOGICO
1	Leao	1	Zoológico de SP
2	Tigre	2	Simba Safári
3	Koala	2	Simba Safári
4	Girafa	1	Zoológico de SP
5	Elefante	2	Simba Safári

Neste caso usamos um ALIAS, ou seja, a coluna receberá o nome ZOOLOGICO;

Esta é uma prática MUITO COMUM;

Da tabela a seguir como trocamos ‘China’ ou ‘Índia’ por ‘Ásia’, ‘Congo’ por ‘África’ e Austrália por ‘Oceania’?

id	nome	zoo	país
1	Leao	1	China
2	Tigre	2	Congo
3	Koala	2	Australia
4	Girafa	1	Congo
5	Elefante	2	India

CASE funciona normalmente para strings também

```
SELECT
  *,
  CASE
    WHEN PAIS = 'China' or PAIS = 'India' THEN 'Asia'
    WHEN PAIS = 'Congo' THEN 'Africa'
    WHEN PAIS = 'Australia' THEN 'Oceania'
    ELSE 'Desconhecido'
  END AS CONTINENTE
FROM ANIMAIS;
```

id	nome	zoo	país	CONTINENTE
1	Leao		1 China	Asia
2	Tigre		2 Congo	Africa
3	Koala		2 Australia	Oceania
4	Girafa		1 Congo	Africa
5	Elefante		2 India	Asia

```
SELECT
  *,
CASE
  WHEN PAIS = 'China' or PAIS = 'India' THEN 'Asia'
  WHEN PAIS = 'Congo' THEN 'Africa'
  WHEN PAIS = 'Australia' THEN 'Oceania'
  ELSE 'Desconhecido'
END AS CONTINENTE
FROM ANIMAIS;
```

Note que quando a informação contida para a coluna é um texto (string), a mesma deverá ser escrita da mesma forma que consta na tabela (conceito similar ao PROC do Excel)

Ex: na tabela consta 'Australia', e não
'Austrália' ou 'AUSTRALIA'



Da tabela a seguir como classificamos cada nome em ‘4 letras’, ‘5 letras’ ou ‘6 letras’?

id	nome
1	Leao
2	Tigre
3	Koala
4	Girafa
5	Elefante

Podemos ainda juntar CASE com LIKE e fazer outras classificações

```
SELECT
  *,
  CASE
    WHEN NOME LIKE '___' THEN '4 Letras'
    WHEN NOME LIKE '____' THEN '5 Letras'
    WHEN NOME LIKE '_____' THEN '6 Letras'
    ELSE 'Mais de 6 letras'
  END AS QTD_LETRAS
FROM ANIMAIS;
```

id	nome	QTD_LETRAS
1	Leao	4 Letras
2	Tigre	5 Letras
3	Koala	5 Letras
4	Girafa	6 Letras
5	Elefante	Mais de 6 Letras



Lembrete:
aprendemos sobre o
operador LIKE e o
wildcard _ na aula 4



Então podemos usar os critérios do filtro ('=' , 'LIKE' , 'BETWEEN' , etc) dentro do CASE WHEN

Traga o nome e o preço da tabela ‘produtos’, mas exiba “Preço desconhecido” para produtos cujo preço for nulo

Traga o nome e o preço de todos os produtos, mas exiba "Preço desconhecido" para produtos cujo preço for nulo

Resposta:

```
SELECT
    nome,
    CASE
        WHEN preco IS NULL THEN 'Preço desconhecido'
        ELSE preco
    END AS preco_formatado
FROM produtos;
```

Selecione o nome e a idade da tabela clientes, mas exiba "Idade desconhecida" para clientes sem uma idade definida:

Selecione o nome e a idade da tabela clientes, mas exiba "Idade desconhecida" para clientes sem uma idade definida:

Resposta:

```
SELECT
    nome,
    CASE
        WHEN idade IS NULL THEN 'Idade desconhecida'
        ELSE idade
    END AS idade_formatada
FROM clientes;
```

O que faz a query abaixo?

```
SELECT
    nome,
    preco,
    CASE
        WHEN preco > 100 THEN preco * 0.9
        ELSE NULL
    END AS preco_com_desconto
FROM produtos;
```

O que faz a query abaixo?

```
SELECT
    nome,
    preco,
    CASE
        WHEN preco > 100 THEN preco * 0.9
        ELSE NULL
    END AS preco_com_desconto
FROM produtos;
```

Resposta:

Traz o nome, preço e uma coluna calculada como preço com desconto, que traz o preço * 0.9 caso o preço seja maior que 100, caso contrario traz nulo.

Da tabela PRODUTOS



O que faz a query abaixo?

```
SELECT
    nome,
    CASE
        WHEN SUM(quantidade) > 100 THEN 'Popular'
        WHEN SUM(quantidade) BETWEEN 50 AND 100 THEN 'Moderado'
        ELSE 'Impopular'
    END AS Popularidade
FROM produtos
GROUP BY nome
```

O que faz a query abaixo?

```
SELECT
    nome,
    CASE
        WHEN SUM(quantidade) > 100 THEN 'Popular'
        WHEN SUM(quantidade) BETWEEN 50 AND 100 THEN 'Moderado'
        ELSE 'Impopular'
    END AS Popularidade
FROM produtos
GROUP BY nome
```

Resposta:

Traz o nome e uma coluna calculada como preço com popularidade, que traz 'Popular' caso a soma de vendas seja maior que 100, 'Moderado' caso a soma de vendas seja entre 50 e 100, caso contrario traz 'Impopular'.



Da tabela PRODUTOS



Olá, precisamos de um relatório que classifique cada aluno em aprovado ou reprovado, a depender da sua nota

Dada a tabela abaixo, calcular a média que cada aluno teve com as 3 provas

id	P1	P2	P3
9651433	2.0	9.0	6.0
5501077	4.0	3.0	5.0
4805256	8.0	9.0	1.0
5334050	4.0	3.0	8.0
4234237	3.0	8.0	0.0
1194269	4.0	7.0	7.0

Dada a tabela abaixo, calcular a média que cada aluno teve com as 3 provas

Resposta:

```
SELECT
    *,
    (P1+P2+P3)/3 AS MEDIA
FROM NOTAS1
```

id	P1	P2	P3	MEDIA
9651433	2.0	9.0	6.0	5.67
5501077	4.0	3.0	5.0	4.0
4805256	8.0	9.0	1.0	6.0
5334050	4.0	3.0	8.0	5.0
4234237	3.0	8.0	0.0	3.67
1194269	4.0	7.0	7.0	6.0

Dada a tabela abaixo, trazer o status, que deve ser 'APROVADO' ou 'REPROVADO'.

Aluno será 'aprovado' se a média for maior ou igual a 5. E 'REPROVADO' caso contrário.

id	P1	P2	P3	MEDIA
9651433	2.0	9.0	6.0	5.67
5501077	4.0	3.0	5.0	4.0
4805256	8.0	9.0	1.0	6.0
5334050	4.0	3.0	8.0	5.0
4234237	3.0	8.0	0.0	3.67
1194269	4.0	7.0	7.0	6.0

Resolução

```
SELECT
    T1.*,
    (P1+P2+P3)/3 AS MEDIA,
    CASE
        WHEN (P1+P2+P3)/3 >= 5 THEN 'APROVADO'
        ELSE 'REPROVADO'
    END AS CLASS_NOTA
FROM NOTAS1 AS T1
```

Note que podemos trazer o cálculo para encontrar a média do aluno, dentro da condição do case when.



O sinal \geq significa “quando o valor for maior ou igual”

Para a seguinte tabela temos as informações das notas e as frequências.

Devemos calcular a média e obter um status:

'APROVADO' se média maior ou igual a 5 e freq for maior que 70%;

'REP. FREQ.' se média maior ou igual a 5 e freq menor que 70%;

'REPROVADO' nos outros casos

id	P1	P2	P3	Freq
9651433	2.0	9.0	6.0	1.0
5501077	4.0	3.0	5.0	0.9
4805256	8.0	9.0	1.0	0.8
5334050	4.0	3.0	8.0	0.5
4234237	3.0	8.0	0.0	0.5
1194269	4.0	7.0	7.0	0.6
3749932	3.0	4.0	4.0	0.9

Resolução

```
SELECT
    T1.*,
    (P1+P2+P3)/3 AS MEDIA,
    CASE
        WHEN (P1+P2+P3)/3 >= 5 AND FREQ >= 0.7 THEN 'APROVADO'
        WHEN (P1+P2+P3)/3 >= 5 AND FREQ < 0.7 THEN 'REP. FREQ.'
        ELSE 'REPROVADO'
    END AS CLASS_NOTA
FROM NOTAS2 AS T1
```



Utilizando o AND podemos adicionar mais uma condição para o mesmo WHEN

Novamente temos uma tabela com as notas e as frequências.

Mas agora temos uma quantidade maior de condições a serem consideradas:

'APROVADO' - quando Nota maior ou igual a 5 e frequência maior que 70%;

'RN' - quando Reprovado por Nota - Nota menor que 5 e frequência maior que 70%;

'RF' - quando Reprovado por Falta - Nota maior ou igual a 5 e frequência menor que 70%;

'RA' - quando Reprovado por Ambos - Nota menor que 5 e frequência menor que 70%;

'REC' - quando Recuperação - Nota entre 3 e 5, frequência maior ou igual que 70%;

'REPROVADO' nos outros casos.

id	P1	P2	P3	Freq
9651433	2.0	9.0	6.0	1.0
5501077	4.0	3.0	5.0	0.9
4805256	8.0	9.0	1.0	0.8
5334050	4.0	3.0	8.0	0.5
4234237	3.0	8.0	0.0	0.5
1194269	4.0	7.0	7.0	0.6
3749932	3.0	4.0	4.0	0.9

Resolução

```
SELECT
    T1.*,
    (P1+P2+P3)/3 AS MEDIA,
    CASE
        WHEN (P1+P2+P3)/3 >= 5 AND FREQ >= 0.7 THEN 'APROVADO'
        WHEN (P1+P2+P3)/3 < 5 AND FREQ >= 0.7 THEN 'RN'
        WHEN (P1+P2+P3)/3 >= 5 AND FREQ < 0.7 THEN 'RF'
        WHEN (P1+P2+P3)/3 < 5 AND FREQ < 0.7 THEN 'RA'
        WHEN (P1+P2+P3)/3 BETWEEN 3 AND 5 AND FREQ >= 0.7 THEN 'REC'
        ELSE 'REPROVADO'
    END AS CLASS_NOTA
FROM NOTAS_DESAFIO AS T1
```

Podemos fazer um CASE sem designar um valor para ELSE? Se sim, que valor ele recebe quando não existir possibilidade?

Podemos fazer um CASE sem designar um valor para ELSE? Se sim, que valor ele recebe quando não existir possibilidade?

Resposta:

Podemos!

Ele recebe NULL (nulo) caso não tenha valor designado.

Desafio: Lista de Clientes

O chefe de vendas da empresa que você trabalha teve a ideia de criar equipes de vendas regionais especializadas que serão capazes de vender scooters para clientes em regiões específicas, em vez de equipes de vendas genéricas.

Para transformar sua ideia em realidade, ele gostaria de ter uma lista de todos os clientes mapeados para regiões.

Para os clientes dos estados de MA, NH, VT, ME, CT ou RI, ele gostaria que fossem rotulados como Nova Inglaterra.

Para os clientes dos estados de GA, FL, MS, AL, LA, KY, VA, NC, SC, TN, VI, WV ou AR, ele gostaria que fossem rotulados como Sudeste.

Os clientes de qualquer outro estado devem ser rotulados como Outro.

Cliente	Estado
Cliente 1	MA
Cliente 2	FL
Cliente 3	NY
Cliente 4	VT
Cliente 5	GA
Cliente 6	CT
Cliente 7	TX
Cliente 8	SC
Cliente 9	NH
Cliente 10	CA

Resposta Desafio: Lista de Clientes

```
SELECT
    Cliente,
    Estado,
    CASE
        WHEN Estado IN ('MA', 'NH', 'VT',
                        'ME', 'CT', 'RI') THEN 'Nova Inglaterra'
        WHEN Estado IN ('GA', 'FL', 'MS',
                        'AL', 'LA', 'KY', 'VA', 'NC', 'SC', 'TN',
                        'VI', 'WV', 'AR') THEN 'Sudeste'
        ELSE 'Outro'
    END AS Regiao
FROM
    clientes;
```

Cliente	Estado	Região
Cliente 1	MA	Nova Inglaterra
Cliente 2	FL	Sudeste
Cliente 3	NY	Outro
Cliente 4	VT	Nova Inglaterra
Cliente 5	GA	Sudeste
Cliente 6	CT	Nova Inglaterra
Cliente 7	TX	Outro
Cliente 8	SC	Sudeste
Cliente 9	NH	Nova Inglaterra
Cliente 10	CA	Outro

Desafio: Como encontrar as linhas na tabela MyTable que possuem exatamente um valor igual a 1 nas colunas f1 até f10?

mytable

keycol	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
1	1	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0

Resposta Desafio: Como encontrar as linhas na tabela MyTable que possuem exatamente um valor igual a 1 nas colunas f1 até f10?

```
SELECT keycol FROM MyTable WHERE  
(f1 + f2 + f3 + f4 + f5 + f6 + f7 + f8 + f9 + f10) = 1  
AND (f1 + f2 + f3 + f4 + f5 + f6 + f7 + f8 + f9 + f10 - (CASE WHEN f1 = 1 THEN 1  
ELSE 0 END) - (CASE WHEN f2 = 1 THEN 1 ELSE 0 END) - (CASE WHEN f3 = 1 THEN 1  
ELSE 0 END) - (CASE WHEN f4 = 1 THEN 1 ELSE 0 END) - (CASE WHEN f5 = 1 THEN 1  
ELSE 0 END) - (CASE WHEN f6 = 1 THEN 1 ELSE 0 END) - (CASE WHEN f7 = 1 THEN 1  
ELSE 0 END) - (CASE WHEN f8 = 1 THEN 1 ELSE 0 END) - (CASE WHEN f9 = 1 THEN 1  
ELSE 0 END) - (CASE WHEN f10 = 1 THEN 1 ELSE 0 END)) = 0
```

CREATE TABLE e INSERT INTO

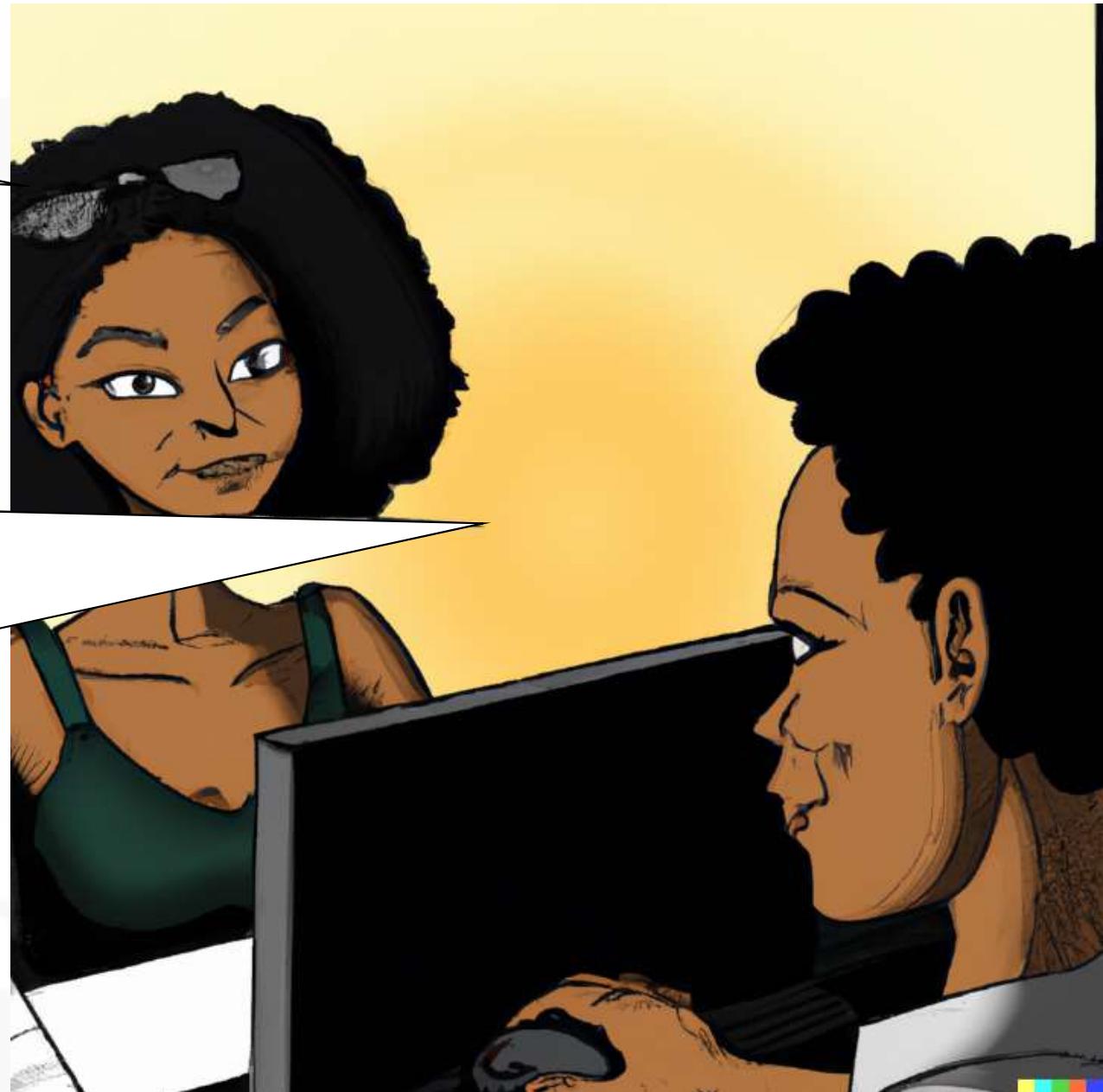
Por que aprender a
criar tabelas?

É um dos comandos que você vai usar o
tempo todo.

Diferentes pessoas de diferentes papéis
numa organização criam tabelas.

Uma prática MUITO comum é **criar**
tabelas resposta do seu trabalho e
tabelas intermediárias para atingir seus
objetivos.

Criar uma tabela pode ser considerada a
resposta final do seu trabalho.



Sou DBA e administro, crio e modifico tabelas para atender às necessidades dos analistas da empresa

Sou cientista de dados e eventualmente armazeno dados de minhas análises em tabelas



Sou estudante de SQL e preciso criar as tabelas para fazer os exercícios propostos no curso

Uma tabela é apenas e tão somente a **combinação de linhas e colunas**.

Em SQL **colunas** podem ser chamadas de **campos**; E **linhas** podem ser chamadas de **registros**.

De forma mais abstrata, cada **tabela** deveria representar uma **entidade** (coisas do mundo real ou relações entre coisas).



Como criar uma tabela?

A sintaxe é:

```
CREATE TABLE NOME_DA_TABELA  
(  
COLUNA1 TIPO1,  
COLUNA2 TIPO2,  
COLUNA3 TIPO3  
...  
);
```

- **CREATE TABLE** seguido do **nome da tabela**
- Dentro dos parênteses temos a definição **do conjunto de colunas** que estarão contidas na tabela criada.
- Cada coluna deve ter seu **tipo** (formato) de dado **definido** aqui. Podemos ter, por exemplo, tipo texto, numérico, data.



Para criar uma tabela o comando é:

CREATE TABLE seguido do nome da tabela

Exemplo de Criação de uma Tabela: CLIENTE

```
CREATE TABLE CLIENTE
(
    ID INTEGER,
    NOME VARCHAR(255),
    DATA_REGISTRO DATE
);
```

A tabela cliente terá os seguintes campos:

ID – que será tipo inteiro (apenas números inteiros, ou seja, sem casa decimal)

NOME – que será string de tamanho máximo 255 (ou seja, máximo de 255 caracteres)

DATA_REGISTRO – que será uma data



Depois do nome da tabela **abrimos parênteses** e colocamos **o nome das colunas desejadas**.

Depois de cada coluna precisamos definir **o tipo de dado** daquela coluna.

Estes são os tipos de dados que podemos usar para criar as colunas no SQLite:

Tipo de Dado	Descrição
INTEGER	Números inteiros
VARCHAR	Strings de tamanho variável para economizar espaço até 255 caracteres
CHAR	Strings de tamanho fixo
FLOAT	Números quebrados/decimais
DECIMAL	É igual o float, mas permite definir o tamanho
TIME	Exemplo: 14:30:52
DATE	Exemplo: 2020-09-09
DATETIME	Exemplo: 2020-09-09 14:30:52
BLOB	Binários grandes



Lembrete: o campo de data geralmente é a data invertida YYYY-MM-DD



São muitos tipos de dados. Para facilitar posso decorar apenas alguns?

Sim, de fato na maior parte das vezes são utilizados apenas os tipos de dados:

- 1- INTEGER
- 2- VARCHAR
- 3- FLOAT
- 4- DATE



Precisamos de uma tabela chamada 'MINHA_TABELA' para armazenar:

- 1- O nome do aluno
- 2- O id do aluno
- 3- A nota do aluno

Exemplo: Criar uma tabela chamada minha_tabela com 3 colunas: *aluno*, *id* e *nota*

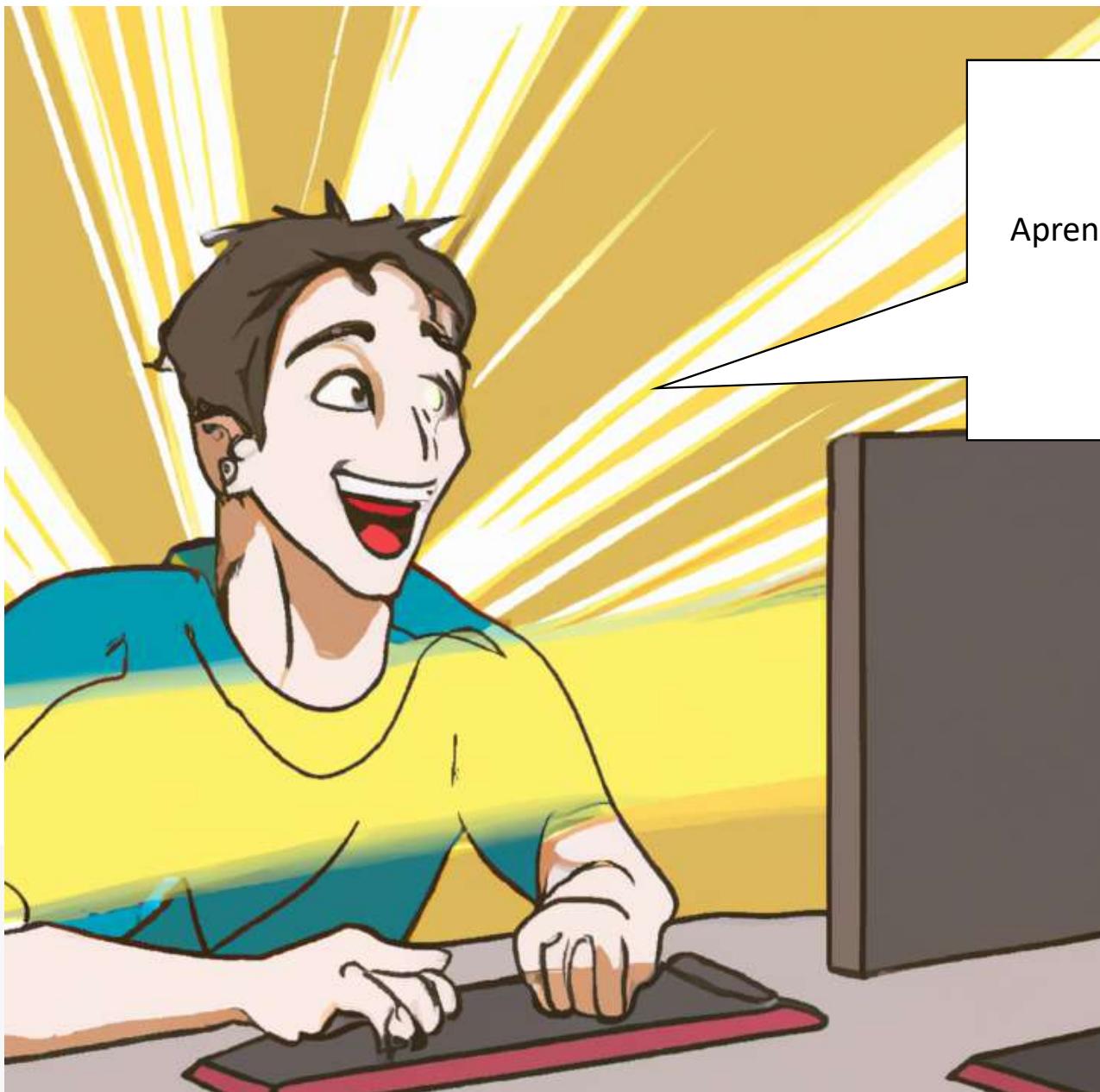
```
CREATE TABLE MINHA_TABELA
(
    ALUNO VARCHAR(255),
    ID INTEGER,
    NOTA INTEGER
);
```



Posso dar o nome
que quiser para a
tabela e para as
colunas da tabela?

Sim, desde que obejam às
seguintes regras:

- 1- Os nomes devem ter
apenas caracteres
alfanuméricos (letras ou
números)
- 2- Os nomes devem começar
apenas com caracteres
alfabéticos (letras) ou '_'



Aprendi a criar minhas primeiras tabelas em
SQL!!



Muito bom! Agora precisamos inserir alguns dados
nesta tabela...

Uma vez criada uma tabela, precisamos inserir dados nela. Isto é feito com o comando INSERT INTO

A sintaxe é:

```
INSERT INTO TABELA VALUES  
(  
    VALOR1,  
    VALOR2,  
    VALOR3,  
    ...  
);
```

INSERT INTO seguido do **nome da tabela** e **VALUES** (**lista de valores separados por vírgula**);

A ordem definida para as colunas da tabela que foi criada DEVE SER a mesma ordem da inserção dos valores dentro do parênteses (depois de VALUES);

Logo, o valor correspondente a COLUNA1 deverá vir primeiro, sendo VALOR1, e assim sucessivamente.

Exemplo:

Inserir na tabela minha_tabela, 5 linhas com os seguintes valores: nomes dos alunos, ids, notas;

```
INSERT INTO MINHA_TABELA VALUES ('Alan',1,10);
INSERT INTO MINHA_TABELA VALUES ('Lorena',2,7);
INSERT INTO MINHA_TABELA VALUES ('Nicole',3,5);
INSERT INTO MINHA_TABELA VALUES ('Tomas',4,8);
INSERT INTO MINHA_TABELA VALUES ('James',5,3);
```



Sempre me confundo. Quais campos precisam de aspas simples? E quais não precisam?

VARCHAR, DATE, CHAR e BLOB **precisam** de aspas simples.

Números ao serem inseridos numa tabela **não precisam** de aspas.



Aprendi a criar minhas primeiras tabelas e
inseri os primeiros valores usando SQL!!



Cuidado!

O INSERT também funciona em tabelas já existentes. O que significa que você pode prejudicar uma tabela da sua empresa caso não preste a atenção devida.

Criar uma tabela com os alunos do curso e sua respectiva idade

Nome: alunos

2 colunas:

- Nome (Varchar(255))
- Idade (Integer)

Criar uma tabela com os alunos do curso e sua respectiva idade

Resposta:

```
CREATE TABLE ALUNOS (
    NOME VARCHAR(255),
    IDADE INTEGER
);
```

Crie uma tabela chamada "produtos" com as colunas "id" (inteiro), "nome" (texto), "preco" (float) e "quantidade" (inteiro)

Crie uma tabela chamada "produtos" com as colunas "id" (inteiro), "nome" (texto), "preco" (float) e "quantidade" (inteiro)

Resposta:

```
CREATE TABLE produtos (
    id INTEGER,
    nome VARCHAR(50),
    preco FLOAT,
    quantidade INTEGER
);
```

Crie uma tabela chamada "clientes" com as colunas "id" (inteiro), "nome" (texto), "email" (texto) e "telefone" (texto)

Crie uma tabela chamada "clientes" com as colunas "id" (inteiro), "nome" (texto), "email" (texto) e "telefone" (texto)

Resposta:

```
CREATE TABLE clientes (
    id INT,
    nome VARCHAR(50),
    email VARCHAR(50),
    telefone VARCHAR(15)
);
```

Crie uma tabela chamada "vendas" com as colunas "id" (inteiro), "id_produto" (inteiro), "data_venda" (data) e "quantidade" (inteiro).

Crie uma tabela chamada "vendas" com as colunas "id" (inteiro), "id_produto" (inteiro), "data_venda" (data) e "quantidade" (inteiro).

Resposta:

```
CREATE TABLE vendas (
    id INT,
    id_produto INT,
    data_venda DATE,
    quantidade INT
);
```

O que faz a query abaixo?

```
CREATE TABLE ENDERE COS (ID INTEGER,  
    NOME_ENDERECO VARCHAR(255),  
    CIDADE VARCHAR(255),  
    ESTADO VARCHAR(255),  
    PAIS VARCHAR(255),  
    CEP INTEGER,  
    DATA_REGISTRO DATE  
)
```

O que faz a query abaixo?

```
CREATE TABLE ENDERECOS (ID INTEGER,  
    NOME_ENDERECHO VARCHAR(255),  
    CIDADE VARCHAR(255),  
    ESTADO VARCHAR(255),  
    PAIS VARCHAR(255),  
    CEP INTEGER,  
    DATA_REGISTRO DATE  
)
```

Resposta:

Cria uma tabela chamada ENDERECOS com as

colunas:

nome_endereço, cidade,
estado, país, cep e
data_registro;

Da tabela abaixo, como inserir uma linha com: nota=8,7, vendas_domesticas=340 milhões e vendas_internacionais = 270 milhões?

Tabela: movie_sales

id_filme	nota	vendas_domesticas	vendas_internacionais
3	7,9	24.000.000	23.000.000
1	8,3	19.000.000	17.000.000
2	7,2	162.000.000	20.000.000

Da tabela abaixo, como inserir uma linha com: nota=8,7, vendas_domesticas=340 milhões e vendas_internacionais = 270 milhões?

Resposta:

```
INSERT INTO MOVIE_SALES VALUES (2, 8.7, 340000000, 270000000);
```

Insira um novo registro na tabela "produtos" com os valores "1" para o ID, "Camiseta" para o nome, 29.99 para o preço e 50 para a quantidade

Insira um novo registro na tabela "produtos" com os valores "1" para o ID, "Camiseta" para o nome, 29.99 para o preço e 50 para a quantidade

Resposta:

```
INSERT INTO produtos VALUES (1, 'Camiseta', 29.99, 50);
```

Insira um novo registro na tabela "clientes" com os valores "1" para o ID, "João Silva" para o nome, "joao.silva@email.com" para o e-mail e "(11) 99999-9999" para o telefone.

Insira um novo registro na tabela "clientes" com os valores "1" para o ID, "João Silva" para o nome, "joao.silva@email.com" para o e-mail e "(11) 99999-9999" para o telefone.

Resposta:

```
INSERT INTO clientes (id, nome, email, telefone)
VALUES (1, 'João Silva', 'joao.silva@email.com', '(11) 99999-9999');
```

Insira um novo registro na tabela "funcionarios" com os valores "1" para o ID, "Ana Santos" para o nome, "Gerente de Vendas" para o cargo e 4500.00 para o salário.

Insira um novo registro na tabela "funcionarios" com os valores "1" para o ID, "Ana Santos" para o nome, "Gerente de Vendas" para o cargo e 4500.00 para o salário.

Resposta:

```
INSERT INTO funcionarios (id, nome, cargo, salario)
VALUES (1, 'Ana Santos', 'Gerente de Vendas', 4500.00);
```

O que faz as queries abaixo?

```
CREATE TABLE PRODUTOS (ID INTEGER,  
    NOME VARCHAR(255),  
    COR VARCHAR(255),  
    CUSTO FLOAT,  
    PRECO FLOAT,  
    TAMANHO FLOAT,  
    PESO FLOAT  
);
```

```
INSERT INTO PRODUTOS (1, 'Capacete Esporte',  
    'Branco', 13.8, 34.99, NULL, NULL);
```

O que faz as queries abaixo?

```
CREATE TABLE PRODUTOS (ID INTEGER,  
    NOME VARCHAR(255),  
    COR VARCHAR(255),  
    CUSTO FLOAT,  
    PRECO FLOAT,  
    TAMANHO FLOAT,  
    PESO FLOAT  
);
```

```
INSERT INTO PRODUTOS (1, 'Capacete Esporte',  
    'Branco', 13.8, 34.99, NULL, NULL);
```

Resposta:

Cria uma tabela chamada PRODUTOS com as colunas:

Nome, cor, custo, preco, tamanho e peso e insere valores nesta tabela;



Qual a diferença se um literal chamado ‘Bob’ for armazenado como CHAR(5) versus VARCHAR(5)?

Qual a diferença se um literal chamado 'Bob' for armazenado como CHAR(5) versus VARCHAR(5)?

Resposta:

A diferença entre a literal 'Bob' armazenada como CHAR(5) versus VARCHAR(5) é que o tipo CHAR(5) armazena exatamente 5 caracteres, preenchendo com espaços em branco se necessário, enquanto o tipo VARCHAR(5) armazena uma sequência de até 5 caracteres sem preenchimento com espaços em branco.

Se tivermos uma coluna classificada como
INTEGER, ela pode eventualmente ter valor NULL?

Se tivermos uma coluna classificada como INTEGER, ela pode eventualmente ter valor NULL?

Resposta:

Sim, um atributo do tipo INTEGER pode ter o valor NULL. O valor NULL representa a ausência de um valor específico.

Fornecer o tipo ideal e um exemplo dos dados abaixo:

Variável	Tipo de Dado	Exemplo
Peso		
Rua		
Aniversário		
Tamanho do Contrato		
Salário		
Gênero		

Fornecer o tipo ideal e um exemplo dos dados abaixo:

Resposta:

Variável	Tipo de Dado	Exemplo
Peso	NUMERIC(5,2) ou FLOAT	72.21
Rua	VARCHAR(n)	“Rua do Limão”
Aniversário	DATE	“1990-05-15”
Tamanho do Contrato	INTEGER	3
Salário	NUMERIC(10,2) ou FLOAT	5000.0
Gênero	CHAR(1)	“M”

Suponha que tenha que criar uma tabela para clientes de um restaurante com as colunas: “Id_Cliente”, “Nome_Cliente”, “Num_Visitas” e “Percent_Desc”. Qual o tipo de dado de cada coluna? Qual coluna seria o identificador único?

Variável	Tipo de Dado	Identificador Único?
Id_Cliente		
Nome_Cliente		
Num_Visitas		
Percent_Desc		

Suponha que tenha que criar uma tabela para clientes de um restaurante com as colunas: “Id_Cliente”, “Nome_Cliente”, “Num_Visitas” e “Percent_Desc”. Qual o tipo de dado de cada coluna? Qual coluna seria o identificador único?

Resposta:

Variável	Tipo de Dado	Identificador Único?
Id_Cliente	INTEGER	SIM
Nome_Cliente	VARCHAR(n)	N
Num_Visitas	INTEGER	N
Percent_Desc	NUMERIC(5,2) ou FLOAT	N

Uma tabela pode existir se não tiver nenhuma linha?

Uma tabela pode existir se não tiver nenhuma linha?

Resposta:

Sim, uma tabela pode não ter nenhuma linha. Nesse caso, a tabela estará vazia.

DELETE



Muitas vezes queremos apagar linhas específicas de uma tabela. Isto é feito com o comando DELETE

A sintaxe é:

```
DELETE FROM TABELA WHERE COLUNA = VALOR_QUE_QUER_QUE_SEJA_APAGADO;
```

DELETE seguido do nome da tabela WHERE condição_filtro;



Acabamos de fechar o departamento 16 e precisamos de alguém para apagar este departamento em nosso banco de dados

Exemplo DELETE

```
DELETE FROM departamentos  
WHERE id_departamento = 16
```

id_departamento	nome_departamento	nome_grupo	nome_categoria
1	Cardiologia	Centro do Coração	Clinica
2	Comunicações	Emergência	Operacional
3	Oncologia	Emergência	Clinica
4	Farmácia	Serviços	Técnica
16	Enfermaria Consolação	Sala	Operacional

Exemplo de DELETE para múltiplas linhas

```
DELETE FROM departamentos  
WHERE id_departamento >= 16
```

id_departamento	nome_departamento	nome_grupo	nome_categoria
1	Cardiologia	Centro do Coração	Clinica
2	Comunicações	Emergência	Operacional
3	Oncologia	Emergência	Clinica
4	Farmácia	Serviços	Técnica
16	Enfermaria Consolação	Sala	Operacional
17	Enfermaria Brooklin	Sala	Operacional
18	Enfermaria Tatuapé	Sala	Operacional
19	Farmácia Consolação	Sala	Operacional
20	Farmácia Brooklin	Sala	Operacional



id_departamento	nome_departamento	nome_grupo	nome_categoria
1	Cardiologia	Centro do Coração	Clinica
2	Comunicações	Emergência	Operacional
3	Oncologia	Emergência	Clinica
4	Farmácia	Serviços	Técnica



A sintaxe do DELETE é muito simples.

Mas **cuidado**, uma vez deletado **não tem como recuperar** o dado de maneira fácil.

A única forma de recuperar é por meio de um *crash recovery* (um backup do banco de dados que é feito normalmente no fim do dia);



Antes de deletar, rode um
SELECT para ver se o que está
querendo apagar é aquilo
mesmo

Da tabela abaixo, como escrever uma query para deletar as linhas cujo ano é menor que 2005?

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Da tabela abaixo, como escrever uma query para deletar as linhas cujo ano é menor que 2005?

Resposta:

```
DELETE FROM FILMES  
WHERE ANO < 2005;
```

Da tabela abaixo, como escrever uma query para deletar as linhas cujo diretor seja Peter Jackson

id_filme	titulo	diretor	ano	tam_min
1	O Poderoso Chefão	Francis Ford Coppola	1972	115
2	Batman: Cavaleiro das Trevas	Christopher Nolan	2008	92
3	Os Sete Samurais	Akira Kurosawa	1954	207
4	A Lista de Schindler	Steven Spielberg	1993	195
5	O Senhor dos Anéis: O Retorno do Rei	Peter Jackson	2003	201
6	Pulp Fiction: Tempo de Violência	Quentin Tarantino	1994	154
7	Forrest Gump: O Contador de Histórias	Robert Zemeckis	1994	142
8	Clube da Luta	David Fincher	1999	139
9	A Origem	Christopher Nolan	2010	148
10	Matrix	Lana Wachowski	1999	136

Da tabela abaixo, como escrever uma query para deletar as linhas cujo diretor seja Peter Jackson

Resposta:

```
DELETE FROM FILMES  
WHERE DIRETOR = 'Peter Jackson';
```

O que faz a queries abaixo?

```
DELETE FROM CLIENTES  
WHERE SOBRENOME LIKE 'S%';
```

```
DELETE FROM CLIENTES  
WHERE ID_CLIENTE IN (10,15,17,21);
```

O que faz a queries abaixo?

```
DELETE FROM CLIENTES  
WHERE SOBRENOME LIKE 'S%';
```

Resposta:

Deleta da tabela CLIENTES as linhas cujo sobrenome começam com S

```
DELETE FROM CLIENTES  
WHERE ID_CLIENTE IN (10,15,17,21);
```

Deleta da tabela CLIENTES as linhas cujo ID_CLIENTE seja 10,15,17 ou 21

Agrupando dados Group By



Você quer tirar o máximo proveito dos seus dados?

Aprender a agrupá-los é uma maneira poderosa de organizar e analisar seus dados.

Com essa habilidade, você pode economizar tempo e energia pois a extração é automática.

Em pouco tempo você se tornará especialista em encontrar os dados de que precisa e analisá-los

As funções normalmente usadas para agrregar são:

- **MAX** – máximo
- **MIN** – mínimo
- **SUM** – soma
- **AVG** – média
- **COUNT** – contagem



A sintaxe do group by

```
SELECT  
    COLUNA_AGREGADORA,  
    SUM(COLUNA_VAI_AGREGAR) AS NOVO_NOME  
FROM TABELA  
GROUP BY COLUNA_AGREGADORA;
```

É preciso definir duas colunas:
1- A coluna agregadora
2- A que será agregada

A agregadora aparece após o SELECT e depois do GROUP BY

A coluna que será agregada aparece após as funções de agregação, neste caso o SUM

Definição das colunas agregadora e agregada

Quando se usa a cláusula GROUP BY em uma consulta SQL, a coluna agregadora é aquela que irá receber os resultados agregados da consulta, como soma, média, contagem, etc.

A coluna agregada é aquela que fornece o valor a ser agregado.

Por exemplo:

```
SELECT
    nome_produto,
    SUM(quantidade_vendida * preco_unitario) AS total_vendas
FROM tabela_vendas
GROUP BY nome_produto;
```

Agregadora = nome_produto

Agregada =
quantidade_vendida *
preco_unitario



Poderia obter a soma do total de pontos por jogador?

Vamos analisar a tabela abaixo:

id	jogador	score
1	Joao	134
2	Tomas	146
3	Lucia	20
4	Tomas	118
5	Tomas	102
6	Lucia	90
7	Lucia	34
8	Joao	122

Ela traz:

O id

O nome do jogador

O score do jogador

Como fazer uma query para trazer a **soma de pontos por jogador?**

Como fazer uma query para trazer a soma de pontos por jogador?

```
SELECT  
    JOGADOR,  
    SUM(SCORE) AS TOTAL_SCORE  
FROM JOGADORES  
GROUP BY JOGADOR;
```

JOGADOR	TOTAL_SCORE
Joao	256
Tomas	366
Lucia	144

Observação: A coluna score tem a informação dos scores dos jogadores, logo, para trazer o total do score, é necessário somar as informações da coluna. E para trazer a informação por jogador, precisamos trazer no select a coluna com a informação do jogador, bem como, agrupar



Como fazer uma query para trazer o menor salário por departamento?

```
SELECT  
    DEPARTAMENTO,  
    MIN(SALARIO) AS SALARIO_MIN  
FROM EMPREGADOS  
GROUP BY DEPARTAMENTO;
```



DEPARTAMENTO	SALARIO_MIN
Marketing	1.200
Finanças	2.300
Producao	1.100



Não se esqueça do GROUP BY que deve ser seguido pela coluna que se deseja agrupar.

Como fazer uma query para trazer o máximo de itens vendidos por ano?

```
SELECT  
    ANO,  
    MAX(ITENS) AS MAX_ITENS  
FROM MERCEARIA  
GROUP BY ANO;
```



ANO	MAX_ITENS
2018	345
2017	123
2019	456

Como fazer uma query para trazer a contagem de pessoas por departamento?

```
SELECT  
    DEPARTAMENTO,  
    COUNT(*) AS CONT_PESSOAS  
FROM EMPREGADOS  
GROUP BY DEPARTAMENTO;
```



DEPARTAMENTO	CONT_PESSOAS
Marketing	5
Finanças	15
Producao	7



Quando usamos COUNT não preciso colocar uma coluna dentro. Posso colocar apenas COUNT(*) .

COUNT(*) versus COUNT(nome_coluna)

COUNT(*) e COUNT(nome_coluna) são **duas formas diferentes** de contar o número de linhas em uma tabela.

COUNT(*) conta o número de linhas na tabela. Ele **conta todas as linhas da tabela**, incluindo as que contêm valores nulos.

COUNT(nome_coluna) **conta o número de linhas não nulas de uma coluna específica**. Em outras palavras, ele conta apenas as linhas que têm um valor para a coluna especificada.

```
SELECT COUNT(*) FROM minha_tabela;
```



```
SELECT COUNT(nome_coluna) FROM minha_tabela;
```

Em resumo, COUNT(*) contará todas as linhas de uma tabela, enquanto COUNT(nome_coluna) contará apenas as linhas onde a coluna especificada não é nula.



O que acontece com
o GROUP BY quando
a coluna agregadora
possui NULOS?

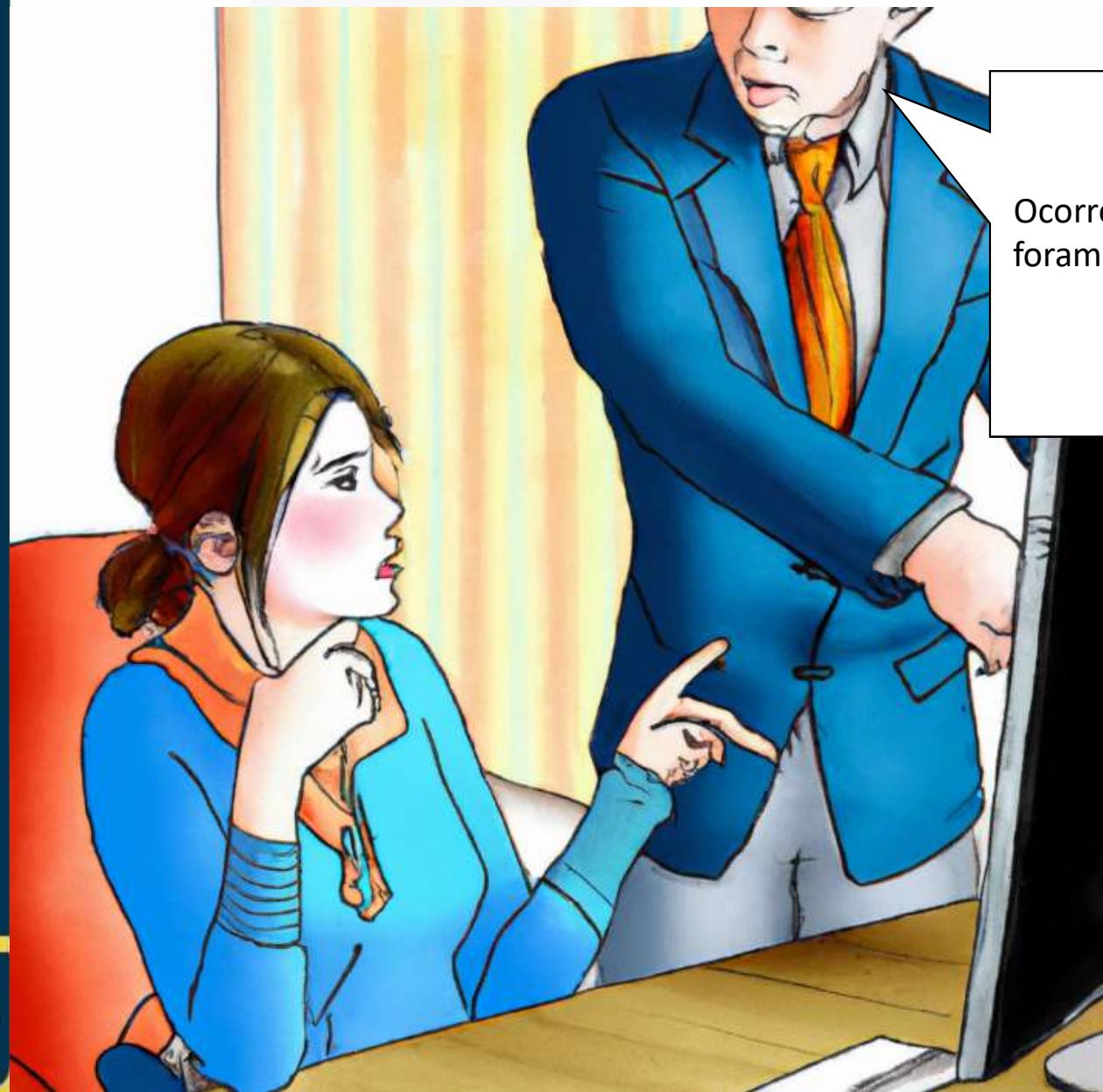
Todas as linhas que
tiverem valor NULL serão
tratadas como uma
agregação diferente, como
veremos no exemplo a
seguir

Vamos analisar a tabela abaixo

id	nome	departamento
1	John Smith	TI
2	Jean	NULL
3	Tomas	Finanças
4	Laura Silva	NULL
5	Ana Madureira	TI
6	Lorena Ferreira	TI

Repare:

Dois departamentos estão NULL (significa que não tiveram informações preenchidas)



Ocorreu um problema na base e alguns funcionários foram adicionados **sem informação preenchida** para o seu departamento

O que acontece se fizermos uma query para trazer a contagem de pessoas por departamento?

```
SELECT  
    DEPARTAMENTO,  
    COUNT(*) AS CONT_PESSOAS  
FROM EMPREGADOS  
GROUP BY DEPARTAMENTO;
```



DEPARTAMENTO	CONT_PESSOAS
NULL	2
TI	3
FINANÇAS	1

NULL entra como uma agregação para departamento e conta quantas linhas estão com departamento NULL



Quando ele vai agragar e tem
algum NULL ele agraga NULL
como se fosse outro valor

A boa prática é trocar NULL por uma string e depois agregar

```
SELECT
    COALESCE(DEPARTAMENTO, 'SEM_DEPARTAMENTO') AS DEPARTAMENTO,
    COUNT(*) AS CONT_PESSOAS
FROM EMPREGADOS
GROUP BY DEPARTAMENTO;
```

DEPARTAMENTO	CONT_PESSOAS
SEM_DEPARTAMENTO	2
TI	3
FINANÇAS	1



Se tivermos NULL na coluna de agregação, e fizermos SUM, AVG, MAX ou MIN, ele vai desconsiderar o NULL no cálculo da soma, máximo ou mínimo. Veja no exemplo a seguir:

Vamos analisar a tabela abaixo

id	nome	departamento	salario
1	John Smith	TI	2000
2	Jean	NULL	2500
3	Tomas	Finanças	2000
4	Laura Silva	NULL	NULL
5	Ana Madureira	TI	1000
6	Lorena Ferreira	TI	NULL

Repare:

Dois salários estão **NULL**

O que acontece se fizermos a média de salário por departamento?

```
SELECT  
    COALESCE(DEPARTAMENTO, 'SEM_DEPARTAMENTO') AS DEPARTAMENTO,  
    AVG(SALARIO) AS SALARIO_MEDIO  
FROM EMPREGADOS  
GROUP BY DEPARTAMENTO;
```

DEPARTAMENTO	SALARIO_MEDIO
SEM_DEPARTAMENTO	2.500
TI	1.500
FINANÇAS	2.000

Valores NULL não entram no cálculo da média

Observação: O nulo que estava na coluna departamento precisa ser tratado, por isso usamos coalesce



Já o nulo do salário é **desconsiderado** no cálculo do salário médio

Traga a soma de vendas de cada id_produto da tabela "vendas"

id	id_produto	quantidade	data_venda
1	1	10	01/03/2022
2	2	15	02/03/2022
3	3	5	02/03/2022
4	1	5	03/03/2022
5	2	10	04/03/2022
6	3	20	05/03/2022
7	1	15	06/03/2022
8	2	5	07/03/2022

Traga a soma de vendas de cada id_produto
da tabela "vendas"

Resposta:

```
SELECT
    id_produto,
    SUM(quantidade) AS total_vendas
FROM vendas
GROUP BY id_produto;
```

Traga a quantidade total de pedidos feitos por cada `id_cliente` da tabela "pedidos"

<code>id</code>	<code>id_cliente</code>	<code>data_pedido</code>
1	1	01/03/2022
2	2	02/03/2022
3	1	02/03/2022
4	3	03/03/2022
5	1	04/03/2022
6	2	05/03/2022
7	2	06/03/2022
8	1	07/03/2022

Traga a quantidade total de pedidos feitos por cada id_cliente da tabela "pedidos"

Resposta:

```
SELECT
    id_cliente,
    COUNT(*) AS total_pedidos
FROM pedidos
GROUP BY id_cliente;
```

Traga o salário médio de cada cargo na tabela "funcionarios"

id	nome	cargo	salario
1	João	Analista	5000.00
2	Maria	Desenvolvedor	6500.00
3	José	Analista	5500.00
4	Ana	Gerente	9000.00
5	Carlos	Desenvolvedor	7000.00
6	Fernanda	Analista	5200.00
7	Rodrigo	Gerente	8500.00
8	Juliana	Desenvolvedor	6000.00
9	Alexandre	Analista	5300.00
10	Paula	Gerente	9200.00

Traga o salário médio de cada cargo na tabela "funcionarios"

Resposta:

```
SELECT
    cargo,
    AVG(salario) AS salario_medio
FROM funcionarios
GROUP BY cargo;
```

Trazer a soma de itens comprados por id_cliente/id_produto:

 id 	 id_cliente 	 id_produto 	 quantidade
1	1	1	10
2	1	2	5
3	2	1	7
4	2	3	12
5	3	2	8
6	3	3	3
7	3	4	6
8	4	2	2
9	4	3	10
10	4	4	4

Trazer a soma de itens comprados por id_cliente/id_produto:

Resposta:

```
SELECT
    id_cliente,
    id_produto,
    SUM(quantidade) AS total_pedidos
FROM pedidos
GROUP BY id_cliente, id_produto;
```

id	id_cliente	id_produto	quantidade
1	1	1	10
2	1	2	5
3	2	1	7
4	2	3	12
5	3	2	8
6	3	3	3
7	3	4	6
8	4	2	2
9	4	3	10
10	4	4	4

Tenho uma sequencia de perguntas para entender melhor quais itens estão vendendo mais em minha livraria. Poderia me ajudar?

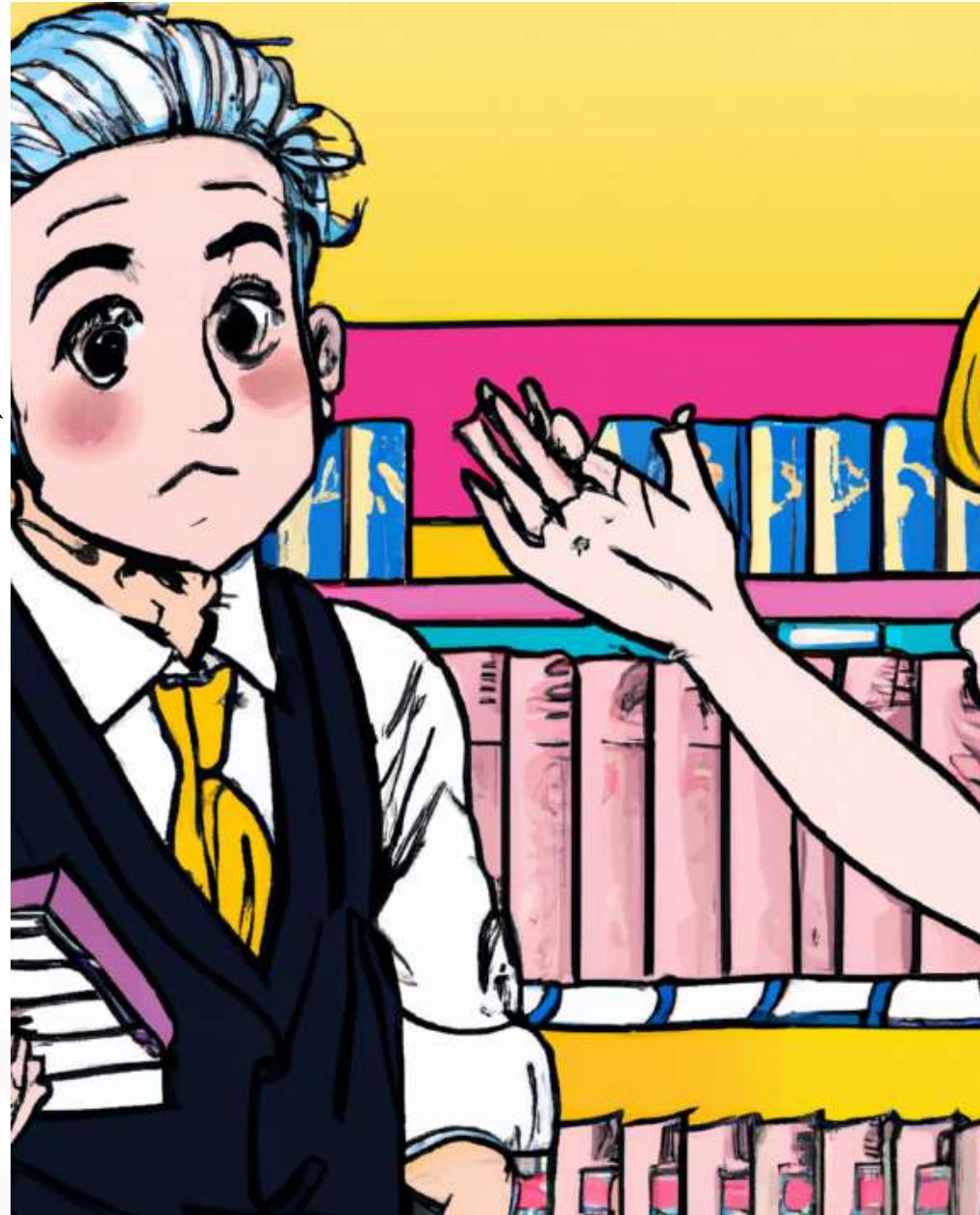
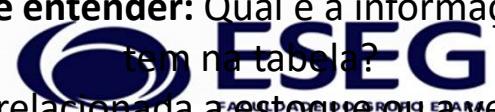


Tabela Livros

id	titulo	autor	genero	idioma	preco	qtd
1	Os 3 mosqueteiros	Alexandre Dumas	aventura	port	11,9	4
2	Game of Thrones	George R. R. Martin	fantasia	ing	8,49	5
3	Orgulho e Preconceito	Jane Austen	romance	port	9,99	2
4	Vampire Academy	Richelle Mead	fantasia	ing	7,99	2
5	Ivanhoe	Walter Scott	aventura	fr	9,99	3
6	Armance	Stendhal	romance	fr	5,99	1

Procure entender: Qual é a informação que tem na tabela?

 Está relacionada a estoque ou a vendas efetuadas?

Escrever uma query que traga o total de vendas por gênero

Tabela Livros

id	titulo	autor	genero	idioma	preco	qtd
1	Os 3 mosqueteiros	Alexandre Dumas	aventura	port	11,9	4
2	Game of Thrones	George R. R. Martin	fantasia	ing	8,49	5
3	Orgulho e Preconceito	Jane Austen	romance	port	9,99	2
4	Vampire Academy	Richelle Mead	fantasia	ing	7,99	2
5	Ivanhoe	Walter Scott	aventura	fr	9,99	3
6	Armance	Stendhal	romance	fr	5,99	1

Resolução

```
SELECT  
    genero,  
    sum(qtd)  
FROM LIVROS  
GROUP BY genero
```

Tabela Resultado

genero	sum(qtd)
aventura	7
fantasia	7
romance	3

Escrever uma query que traga a média de vendas por gênero

Tabela Livros

id	titulo	autor	genero	idioma	preco	qtd
1	Os 3 mosqueteiros	Alexandre Dumas	aventura	port	11,9	4
2	Game of Thrones	George R. R. Martin	fantasia	ing	8,49	5
3	Orgulho e Preconceito	Jane Austen	romance	port	9,99	2
4	Vampire Academy	Richelle Mead	fantasia	ing	7,99	2
5	Ivanhoe	Walter Scott	aventura	fr	9,99	3
6	Armance	Stendhal	romance	fr	5,99	1

Resolução

```
SELECT  
    genero,  
    avg(qtd)  
FROM LIVROS  
GROUP BY genero
```

Tabela Resultado

genero	avg(qtd)
aventura	3,5
fantasia	3,5
romance	1,5

Escrever uma query que traga numa tabela só: o mínimo, o máximo e a média por gênero

Tabela Livros

id	titulo	autor	genero	idioma	preco	qtd
1	Os 3 mosqueteiros	Alexandre Dumas	aventura	port	11,9	4
2	Game of Thrones	George R. R. Martin	fantasia	ing	8,49	5
3	Orgulho e Preconceito	Jane Austen	romance	port	9,99	2
4	Vampire Academy	Richelle Mead	fantasia	ing	7,99	2
5	Ivanhoe	Walter Scott	aventura	fr	9,99	3
6	Armance	Stendhal	romance	fr	5,99	1

Resolução

```
SELECT  
    genero,  
    min(qtd),  
    max(qtd),  
    avg(qtd)  
FROM LIVROS  
GROUP BY genero
```

Tabela Resultado

genero	min(qtd)	max(qtd)	avg(qtd)
aventura	3	4	3,5
fantasia	2	5	3,5
romance	1	2	1,5

Escrever uma query que traga a contagem de títulos por: gênero, idioma

Tabela Livros

id	titulo	autor	genero	idioma	preco	qtd
1	Os 3 mosqueteiros	Alexandre Dumas	aventura	port	11,9	4
2	Game of Thrones	George R. R. Martin	fantasia	ing	8,49	5
3	Orgulho e Preconceito	Jane Austen	romance	port	9,99	2
4	Vampire Academy	Richelle Mead	fantasia	ing	7,99	2
5	Ivanhoe	Walter Scott	aventura	fr	9,99	3
6	Armance	Stendhal	romance	fr	5,99	1

Resolução

```
SELECT
    genero,
    idioma,
    count(*)
FROM LIVROS
GROUP BY genero, idioma
```

Tabela Resultado

genero	idioma	count(*)
aventura	port	1
fantasia	ing	2
romance	port	1
aventura	fr	1
romance	fr	1

Tenho uma sequencia de perguntas para entender melhor quais itens estão vendendo mais em minha concessionária. Poderia me ajudar?



Descrever as informações que existem na prévia da tabela carros abaixo

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branco	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelho	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branco	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viúvo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelho	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Escrever uma query que traga quantos carros foram vendidos por mês

Tabela Origem

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branco	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelho	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branco	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viudo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelho	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Tabela Resultado

Mês	count(*)
Abril	37
Agosto	49
Fevereiro	24
Janeiro	24
Julho	56
Junho	66
Março	31
Maio	48
Outubro	29
Setembro	43

Resolução

```
SELECT  
    MES as Mês,  
    count(*)  
FROM CARROS  
GROUP BY MES
```

Escrever uma query que traga quantos carros foram vendidos de cada cor

Tabela Origem

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branc	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelh	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branc	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viudo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelh	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Tabela Resultado

Cor	count(*)
Prata	104
Branc	75
Vermelh	62
Verde	58
Azul	53
Amarelo	26
Cinza	29

Resolução

```
SELECT
    Cor,
    count(*)
FROM CARROS
GROUP BY Cor
```

Escrever uma query que traga quantos carros foram vendidos de cada modelo

Tabela Origem

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branco	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelho	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branco	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viudo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelho	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Tabela Resultado

Modelo	count(*)
4x4	36
Hatchback	132
Luxury	71
Sedan	116
Sports	38
Supercar	14

Resolução

```
SELECT  
    Modelo,  
    count(*)  
FROM CARROS  
GROUP BY Modelo
```

Escrever uma query que traga quantos carros foram vendidos para solteiros

Tabela Origem

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branco	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelho	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branco	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viuvo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelho	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Tabela Resultado

Estado Civil count(*)	
Casado	304
Solteiro	67
Viuvo	36

Resolução

```
SELECT  
    estado_civil as Estado Civil,  
    count(*)  
FROM CARROS  
GROUP BY estado_civil
```

Escrever uma query que traga o modelo de carro mais vendido para solteiros

Tabela Origem

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branco	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelho	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branco	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viudo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelho	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Tabela Resultado

Estado Civil	Modelo	count(*)
Solteiro	4x4	7
Solteiro	Hatchback	12
Solteiro	Luxury	6
Solteiro	Sedan	9
Solteiro	Sports	28
Solteiro	Supercar	5

Resolução

```
SELECT
    estado_civil as Estado Civil,
    modelo,
    count(*)
FROM CARROS
GROUP BY estado_civil, modelo
```

Escrever uma query que identifique o mês de maior receita

Tabela Origem

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branco	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelho	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branco	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viudo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelho	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Tabela Resultado

Mês	sum(Valor de Venda)
Abril	R\$ 985.389
Agosto	R\$ 1.267.189
Fevereiro	R\$ 657.311
Janeiro	R\$ 572.041
Julho	R\$ 1.547.765
Junho	R\$ 1.606.547
Março	R\$ 845.666
Maio	R\$ 1.114.421
Outubro	R\$ 837.234
Setembro	R\$ 1.098.576

Resolução

```
SELECT  
    mes as Mês,  
    sum(Valor_de_venda)  
FROM CARROS  
GROUP BY mes
```

Escrever uma query que identifique o mês de maior lucro

Tabela Origem

Nome	Idade	Estado Civil	Filhos	Profissão	Casa	Modelo	Cor	Renda	Mês	Valor de Venda	Valor de Compra
Ana	37	Casado	2	Dono de casa	Propria	4x4	Prata	Media	Abril	R\$ 10.039	R\$ 7.529
Bruno	39	Casado	1	Profissional Liberal	Alugada	4x4	Branco	Muito Baixa	Abril	R\$ 13.060	R\$ 9.795
Carlos	29	Casado	1	Administrativo	Propria	4x4	Vermelho	Media	Abril	R\$ 13.869	R\$ 10.402
Daniel	43	Solteiro	0	Empresario	Propria	4x4	Branco	Alta	Abril	R\$ 11.051	R\$ 8.288
Eduardo	39	Viudo	1	Administrativo	Propria	4x4	Prata	Alta	Agosto	R\$ 12.573	R\$ 9.430
Fernanda	34	Casado	2	Dono de casa	Propria	4x4	Verde	Media	Agosto	R\$ 13.851	R\$ 10.388
Gabriel	39	Solteiro	0	Profissional Liberal	Alugada	4x4	Azul	Alta	Fevereiro	R\$ 12.363	R\$ 9.272
Hélia	36	Casado	2	Profissional Liberal	Propria	4x4	Amarelo	Muito Baixa	Janeiro	R\$ 13.161	R\$ 9.871
Igor	38	Casado	2	Administrativo	Propria	4x4	Vermelho	Alta	Janeiro	R\$ 12.178	R\$ 9.134

Tabela Resultado

Mês	sum(Valor de Venda-Valor de Compra)
Abril	R\$ 524.973
Agosto	R\$ 676.067
Fevereiro	R\$ 399.048
Janeiro	R\$ 290.261
Julho	R\$ 807.724
Junho	R\$ 797.577
Março	R\$ 424.387
Maio	R\$ 623.871
Outubro	R\$ 412.885
Setembro	R\$ 591.903

Resolução

```
SELECT  
    mes as Mês,  
    sum(Valor_de_venda - Valor_de_compra)  
FROM CARROS  
GROUP BY mes
```

Se tivermos uma coluna, X, com valores 1, 2, 3 e NULL, qual é a média (AVG) de X?

Se tivermos uma coluna, X, com valores 1, 2, 3 e NULL, qual é a média (AVG) de X?

Resposta:

Se tivermos uma coluna X com valores 1, 2, 3 e NULL, a média (AVG) de X será calculada considerando apenas os valores não nulos. Portanto, a média de X seria $(1 + 2 + 3) / 3 = 2$.

A seguinte expressão:

$$AVG(price) = SUM(price)/COUNT(*)$$

Sempre será verdadeira? Por quê?

A seguinte expressão:

$$AVG(price) = SUM(price)/COUNT(*)$$

Sempre será verdadeira? Por quê?

Resposta:

AVG(price) não é necessariamente igual a SUM(price) / COUNT(). Isso ocorre porque AVG(price) considera apenas os valores não nulos, enquanto SUM(price) / COUNT() considera todos os valores, incluindo NULL. Para corrigir, podemos usar a função AVG com a cláusula DISTINCT para garantir que apenas os valores distintos sejam considerados na média. Por exemplo: AVG(DISTINCT price).

Se tivermos uma coluna X, quando $\text{MIN}(X) = \text{MAX}(X)$ para todas as linhas?

Se tivermos uma coluna X, quando $\text{MIN}(X) = \text{MAX}(X)$ para todas as linhas?

Resposta:

$\text{MIN}(X) = \text{MAX}(X)$ ocorre quando todos os valores na coluna X são iguais. Nesse caso, o valor mínimo e o valor máximo seriam o mesmo.

Se X for uma coluna de chave primária, ou seja, só tem valores únicos, conceitualmente qual a resposta de COUNT(X)? Como esse valor se compara a COUNT(*) na mesma tabela?

Se X for uma coluna de chave primária, ou seja, só tem valores únicos, conceitualmente qual a resposta de COUNT(X)? Como esse valor se compara a COUNT(*) na mesma tabela?

Resposta:

Se X é uma chave primária, COUNT(X) retornará o número total de linhas na tabela. Será equivalente a COUNT(*) sobre a mesma tabela, pois ambas contam o número de linhas na tabela.

WHERE se refere a linhas assim como HAVING
se refere a ... ?

WHERE se refere a linhas assim como HAVING
se refere a ... ?

Resposta:

WHERE é para linhas assim como HAVING é para grupos.

Desafio: Insights sobre as vendas

O CEO, COO e CFO da ZoomZoom gostariam de obter insights sobre o que pode estar impulsionando as vendas.

Agora finalmente a empresa sente que possui uma equipe de análise forte o suficiente com a sua chegada.

A tarefa foi atribuída a você, e seu chefe educadamente informou que este projeto é o mais importante em que a equipe de análise já trabalhou.

Desafio: Insights sobre as vendas

1. Calcule o número total de vendas de unidades que a empresa realizou.
2. Calcule o valor total de vendas em dólares para cada estado.
3. Identifique as cinco melhores concessionárias em termos de maior número de unidades vendidas (ignore as vendas pela internet).
4. Calcule o valor médio de vendas para cada canal, conforme visto na tabela de vendas, e analise o valor médio de vendas primeiro por canal de vendas, depois por `product_id` e, em seguida, por ambos juntos.

Tabela Vendas

venda_id	unidade_venda	estado	concessionaria_id	valor_venda
1		1 CA	1001	1000
2		2 CA	1001	2000
3		1 NY	1002	1500
4		1 NY	1003	1200
5		3 TX	1004	3000
6		2 TX	1004	2500
7		1 FL	1005	1800
8		1 FL	1005	1600
9		2 CA	1001	2200
10		1 NY	1002	1100

Resposta Desafio: Insights sobre as vendas

```
SELECT  
    SUM(unidade_venda) AS total_unidades_vendidas  
FROM vendas;
```

total_unidades_vendidas

13

Resposta Desafio: Insights sobre as vendas

```
SELECT
    estado,
    SUM(valor_venda) AS total_vendas
FROM vendas
GROUP BY estado;
```

estado	total_vendas
CA	7.400
NY	3.800
TX	5.500
FL	3.400

Resposta Desafio: Insights sobre as vendas

```
SELECT
    concessionaria_id,
    SUM(unidade_venda) AS total_unidades_vendidas
FROM vendas
WHERE concessionaria_id IS NOT NULL
GROUP BY concessionaria_id
ORDER BY total_unidades_vendidas DESC
LIMIT 5;
```

concessionaria_id	total_unidades_vendidas
1001	4
1004	5
1002	2
1005	2
1003	1

Resposta Desafio: Insights sobre as vendas

```
-- Por canal de vendas
SELECT
    canal_vendas,
    AVG(valor_venda) AS media_vendas
FROM vendas
GROUP BY canal_vendas;

-- Por product_id
SELECT
    product_id,
    AVG(valor_venda) AS media_vendas
FROM vendas
GROUP BY product_id;

-- Por canal de vendas e product_id
SELECT
    canal_vendas,
    product_id,
    AVG(valor_venda) AS media_vendas
FROM vendas
GROUP BY canal_vendas, product_id;
```

canal_vendas	media_vendas
Online	2.000
Loja	15.333.333
Telefone	1.985

product_id	media_vendas
1	1.000
2	2.400
3	1.400

canal_vendas	product_id	media_vendas
Online	1	1.000
Loja	1	1.000
Telefone	1	1.100
Online	2	2.000
Loja	2	2.700
Telefone	2	2.300
Online	3	1.200
Loja	3	1.600
Telefone	3	1.400

Desafio: Como encontrar o valor mais comum (MODA) de valor pago (check_amt) e a quantidade de vezes que ele aparece na tabela de folha de pagamento abaixo?

payroll

check_nbr	check_amt
1	100.00
2	150.00
3	100.00
4	75.00
5	200.00
6	100.00
7	100.00
8	50.00
9	100.00
10	150.00

Resposta Desafio: Como encontrar o valor mais comum (MODA) de valor pago (check_amt) e a quantidade de vezes que ele aparece na tabela de folha de pagamento abaixo?

```
SELECT check_amt, COUNT(check_amt) as occurrences
FROM Payroll
GROUP BY check_amt
ORDER BY occurrences DESC
LIMIT 1;
```

Desafio: Como apresentar os dados de forma que mostre os dias de processamento, conclusão e confirmação de cada pedido (order_nbr) para um determinado tipo de serviço (service_type)?

serviceschedule

shop_id	order_nbr	sch_seq	service_type	sch_date
002	4155526710	1	01	'1994-07-16'
002	4155526710	2	01	'1994-07-30'
		3	01	'1994-10-01'
003	4155526711	1	01	'1994-07-16'
		2	01	'1994-07-30'
003	4155526712	1	01	'1994-07-16'
		2	01	'1994-07-30'
		3	01	'1994-10-01'
004	4155526713	1	01	'1994-07-16'
		2	01	'1994-07-30'

Resposta Desafio: Como apresentar os dados de forma que mostre os dias de processamento, conclusão e confirmação de cada pedido (order_nbr) para um determinado tipo de serviço (service_type)?

```
SELECT order_nbr,
       MAX(CASE WHEN sch_seq = 1 THEN sch_date ELSE NULL END) AS processed,
       MAX(CASE WHEN sch_seq = 2 THEN sch_date ELSE NULL END) AS completed,
       MAX(CASE WHEN sch_seq = 3 THEN sch_date ELSE NULL END) AS confirmed
  FROM ServicesSchedule
 WHERE service_type = 'e1'
 GROUP BY order_nbr
```

Desafio: Como escrever uma query para encontrar os nomes dos pilotos que podem voar todos os aviões no hangar?

pilot

pilot	plane
John Smith	Piper Cub
Jane Doe	B-52 Bomber
Bob Johnson	F-14 Fighter
Michael Lee	Piper Cub
Emily Davis	B-52 Bomber
David Wilson	F-14 Fighter
Daniel Brown	B-1 Bomber
Matthew Taylor	B-52 Bomber
Ryan Harris	F-14 Fighter
Joseph Garcia	B-1 Bomber

Resposta Desafio: Como escrever uma query para encontrar os nomes dos pilotos que podem voar todos os aviões no hangar?

```
SELECT pilot
FROM Pilotskills
GROUP BY pilot
HAVING COUNT(DISTINCT plane) = (SELECT COUNT(*) FROM Hangar)
```

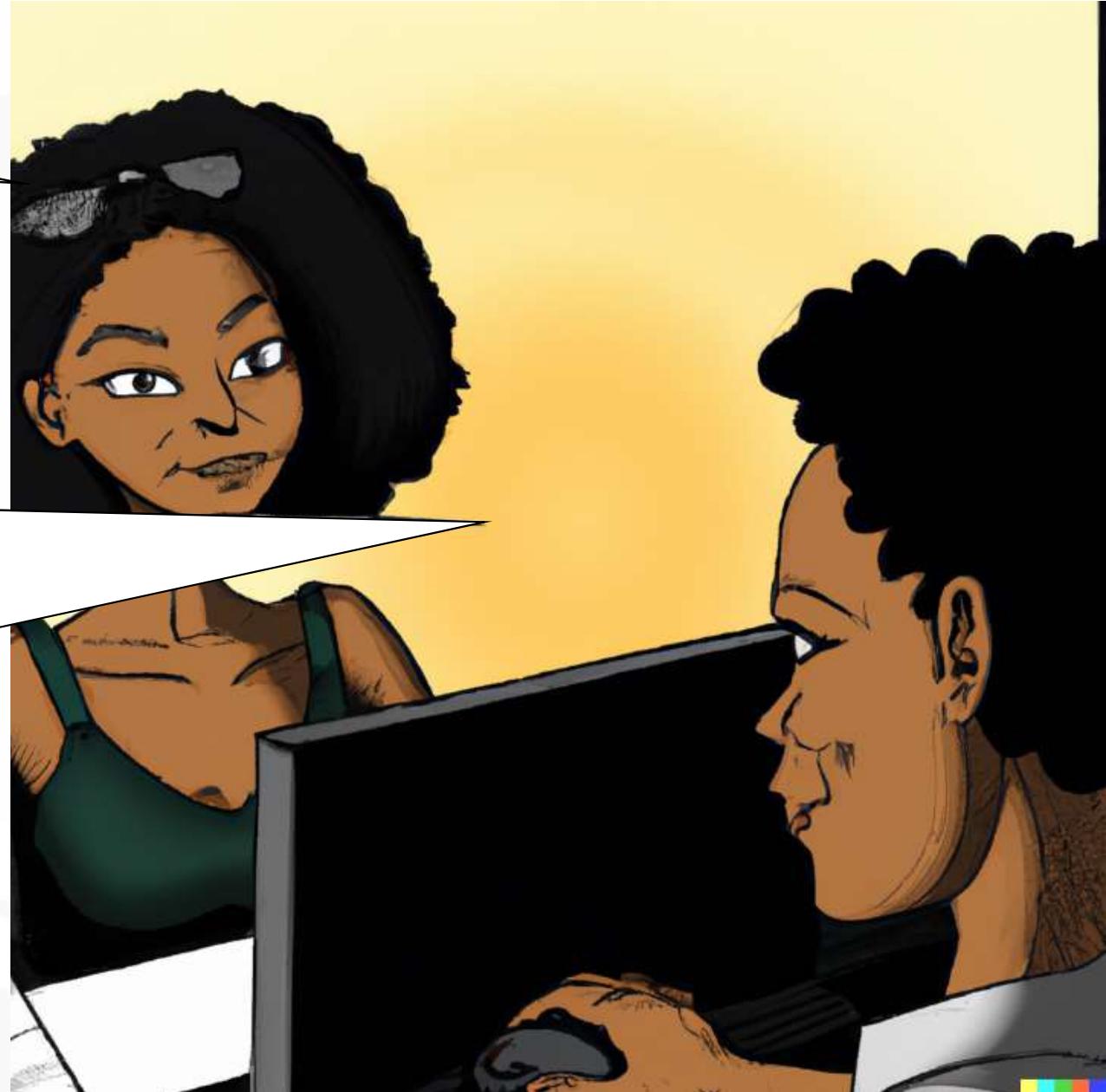
Juntando Tabelas

Por que aprender a juntar tabelas?

É um dos comandos que você vai usar o tempo todo.

Na prática nos bancos de dados temos muitas tabelas

Muitas vezes você quer juntar duas ou mais tabelas na vertical ou na horizontal



Existem duas formas de juntar tabelas:

Usando o **JOIN** e usando o **UNION**



Para juntar duas ou mais tabelas na horizontal usamos o JOIN

Suponha que temos duas tabelas: a tabela 1 e a tabela2

Tabela 1

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Tabela 2

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Podemos juntá-las na horizontal...

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100



Para juntar as tabelas na horizontal usamos o JOIN

Para juntar duas ou mais tabelas na vertical usamos o UNION

Tabela 1

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Tabela 2

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Ou podemos juntá-las na vertical...

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Para juntar as tabelas na vertical usamos o UNION



JOIN



Sintaxe do JOIN

```
SELECT
  *
FROM tabela1
LEFT JOIN tabela2
ON tabela1.id1 = tabela2.id2
```

OU

```
SELECT
  *
FROM tabela1
INNER JOIN tabela2
ON tabela1.id1 = tabela2.id2
```

Parte de um SELECT SIMPLES na tabela1

Seguido de algum JOIN na tabela2

ON define a chave de junção

tabela1 e tabela2 são os nomes das tabelas;
As colunas são id1 e id2

Ou seja, vamos juntar a coluna id1 da tabela1 na id2 da tabela2;



É possível juntar
mais de uma tabela
de uma vez só?

Sim e este é um
procedimento muito
utilizado.

Veremos como se faz
mais a seguir.

Um primeiro exemplo de uso do JOIN

id	titulo_musica	id_artista
1	Dancing queen	4
2	Imagine	2
3	I will survive	1
4	Rolling in the deep	6
5	Respect	3
6	Mamma mia	4

id_artista	nome_artista
1	Gloria Gaynor
2	John Lennon
3	Aretha Franklin
4	ABBA
5	Michael Jackson

A tabela 1 tem o id da música, o nome da música e o id do artista;
A tabela 2 tem o id do artista e o nome do artista;

Como vamos juntar a tabela1 com a tabela2?

Em outras palavras: quais colunas serão as chaves de junção?

Exemplo para Left Join

```
SELECT
  *
FROM musicas
LEFT JOIN artistas
ON musicas.id_artista = artistas.id_artista
```

id	titulo_musica	id_artista	id_artista	nome_artista
1	Dancing queen	4	4	ABBA
2	Imagine	2	2	John Lennon
3	I will survive	1	1	Gloria Gaynor
4	Rolling in the deep	6	NULL	NULL
5	Respect	3	3	Aretha Franklin
6	Mamma mia	4	4	ABBA

No **left join** trazemos **TODOS os artistas que estão na tabela1**; Se não tiver aquele artista na tabela2, traz-se o valor **NULL**;

Exemplo para Inner Join

```
SELECT
*
FROM musicas
INNER JOIN artistas
ON musicas.id_artista = artistas.id_artista
```

<u>id</u>	<u>titulo_musica</u>	<u>id_artista</u>	<u>id_artista</u>	<u>nome_artista</u>
1	Dancing queen	4	4	ABBA
2	Imagine	2	2	John Lennon
3	I will survive	1	1	Gloria Gaynor
5	Respect	3	3	Aretha Franklin
6	Mamma mia	4	4	ABBA

O INNER JOIN também junta as tabelas, **mas** como tabela resultado traz **apenas as linhas que estão em comum** nas duas tabelas originais;

Left Join versus Inner Join

```
SELECT
  *
FROM musicas
LEFT JOIN artistas
ON musicas.id_artista = artistas.id_artista
```

id	titulo_musica	id_artista	id_artista	nome_artista
1	Dancing queen	4	4	ABBA
2	Imagine	2	2	John Lennon
3	I will survive	1	1	Gloria Gaynor
4	Rolling in the deep	6	NULL	NULL
5	Respect	3	3	Aretha Franklin
6	Mamma mia	4	4	ABBA

```
SELECT
  *
FROM musicas
INNER JOIN artistas
ON musicas.id_artista = artistas.id_artista
```

id	titulo_musica	id_artista	id_artista	nome_artista
1	Dancing queen	4	4	ABBA
2	Imagine	2	2	John Lennon
3	I will survive	1	1	Gloria Gaynor
5	Respect	3	3	Aretha Franklin
6	Mamma mia	4	4	ABBA

Repare como o INNER JOIN traz APENAS os artistas que estão em comum tanto na tabela de música quanto na tabela de artista;





O resultado do **INNER JOIN** traz apenas as **linhas em comum** nas duas tabelas.

Já o resultado do **LEFT JOIN** traz **todas as linhas da primeira tabela**, independente de existir um correspondente na segunda tabela.



Existem outros JOINS
além do INNER JOIN
e o LEFT JOIN?

Sim mas por enquanto
concentre-se nestes
dois, pois são os mais
utilizados
normalmente.

Olá, estou trabalhando com a reposição de estoques

Será que poderia nos dizer quais itens não temos em estoque e precisaremos comprar?



Estudo de Caso: Mercearia

produtos

id_produto	produto
1	suco de laranja
2	pão
3	café
4	ovo
5	cereais
6	chocolate

estoques

id_produto	quantidade
1	7
2	4
4	1
5	13
7	35

A tabela 1 tem o nome do produto e o id do produto (é uma tabela de cadastro);

A tabela 2 tem o id do produto e a quantidade disponível em estoque;

São produtos cadastrados e estocados numa mercearia;

Quais produtos não temos em estoque e vamos precisar comprar?

Antes de responder a essa pergunta precisamos decidir:

- 1- Qual JOIN usaremos? O INNER JOIN ou o LEFT JOIN?
- 2- Qual tabela será a primeira e qual tabela será a segunda no JOIN?





Vejamos... Como na tabela estoques não temos estoque ZERO, para identificar itens que não temos em estoque precisamos ver os itens que temos na tabela produtos mas que não estão presentes na tabela de estoques.

Isto só poder ser feito com o LEFT JOIN.

Mas ainda precisamos responder: qual tabela vem primeiro dentro do JOIN. A tabela produtos ou a tabela estoques?

produtos

id_produto **produto**

1	suco de laranja
2	pao
3	café
4	ovo
5	cereais
6	chocolate

estoques

id_produto **quantidade**

1	7
2	4
4	1
5	13
7	35



Vamos começar com a tabela produtos e ver o que acontece

Utilizando o LEFT JOIN e colocando primeiro a tabela produtos, temos:

```
SELECT
  *
FROM produtos
LEFT JOIN estoques
ON produtos.id_produto = estoques.id_produto
```

id_produto	produto	id_produto	quantidade
1	suco de laranja	1	7
2	pao	2	4
3	café	NULL	NULL
4	ovo	4	1
5	cereais	5	13
6	chocolate	NULL	NULL

A tabela resultado traz a lista com todos os produtos e a quantidade daqueles produtos em estoque.

Para produtos que não temos em estoque a tabela resultante traz nulos.

Agora vamos começar com a tabela estoques e ver o que acontece

Utilizando o LEFT JOIN e colocando primeiro a tabela estoques, temos:

```
SELECT
  *
FROM estoques
LEFT JOIN produtos
ON produtos.id_produto = estoques.id_produto
```

id_produto	quantidade	id_produto	produto
1	7	1	suco de laranja
2	4	2	pao
4	1	4	ovo
5	13	5	cereais
7	35	NULL	NULL

A tabela resultado traz a lista com todos os produtos que temos estoque seguido do nome do produto.

Mas repare ainda que podemos ter itens com estoque mas que não existem na tabela produtos



Analizando as tabelas resposta para entender o que cada uma delas responde:

id_produto	produto	id_produto	quantidade
1	suco de laranja	1	7
2	pao	2	4
3	café	NULL	NULL
4	ovo	4	1
5	cereais	5	13
6	chocolate	NULL	NULL



Partindo da tabela produtos primeiro temos todos os produtos cadastrados seguido da quantidade de estoques.

Para itens que não estão na tabela de estoques o estoque é nulo.

Assim identificamos por exemplo que café e chocolate não tem estoque.

id_produto	quantidade	id_produto	produto
1	7	1	suco de laranja
2	4	2	pao
4	1	4	ovo
5	13	5	cereais
7	35	NULL	NULL



Partindo da tabela de estoques primeiro temos todos os estoques seguido do nome do produto.

Podem haver itens que tem estoque e não nome cadastrado. Esses itens aparentemente não foram cadastrados na tabela produtos.



E mais importante para a análise. Não identificamos que café e chocolate não temos estoque.



A ordem em que coloco as tabelas no JOIN faz
toda a diferença

Descrever as informações que cada uma das tabelas abaixo possui. E se fosse para juntar as duas tabelas, qual seria a coluna de junção?

Cliente

id_cliente	nome_cliente	cidade_cliente	id_vendedor
1	Rivaldo	São Paulo	3
2	José	São Paulo	2
3	Ronaldo	São Paulo	1
4	Adalberto	Rio de Janeiro	2
5	Laura	São Paulo	4
6	Elaine	Maranhão	6
7	Fabiano	São Paulo	7
8	Jeff	Curitiba	1
9	Alberto	São Paulo	5
10	Romualdo	São Paulo	1

Vendedores

id_vendedor	nome_vendedor	cidade_vendedor
1	Joao	São Paulo
2	Gilberto	Rio de Janeiro
3	Sara	São Paulo
4	Renata	Curitiba
5	Jaqueline	João Pessoa
6	Reinaldo	Maranhão

Como escrever uma query que junta as duas tabelas e traga todas as colunas presentes em ambas as tabelas?

Obs: Apenas dos clientes que tem algum vendedor

Tabela Resposta

id_cliente	nome_clie	cidade_cliente	id_vendedor	id_vendedor	nome_ver	cidade_vende
1	Rivaldo	São Paulo	3	3	Sara	São Paulo
2	José	São Paulo	2	2	Gilberto	Rio de Janeiro
3	Ronaldo	São Paulo	1	1	Joao	São Paulo
4	Adalberto	Rio de Janeiro	2	2	Gilberto	Rio de Janeiro
5	Laura	São Paulo	4	4	Renata	Curitiba
6	Elaine	Maranhão	6	6	Reinaldo	Maranhão
8	Jeff	Curitiba	1	1	Joao	São Paulo
9	Alberto	São Paulo	5	5	Jaqueline	João Pessoa
10	Romualdo	São Paulo	1	1	Joao	São Paulo

Como escrever uma query que junta as duas tabelas e traga todas as colunas presentes em ambas as tabelas?

Obs: Apenas dos clientes que tem algum vendedor

Resposta:

```
select
    *
from cliente as t1
inner join vendedores as t2
on t1.id_vendedor = t2.id_vendedor
```

Como escrever uma query para trazer *apenas o nome dos vendedores e a sua respectiva lista de clientes?*

Tabela Resposta

nome_vendedor	nome_cliente
Joao	Jeff
Joao	Romualdo
Joao	Ronaldo
Gilberto	Adalberto
Gilberto	José
Sara	Rivaldo
Renata	Laura
Jaqueline	Alberto
Reinaldo	Elaine

Cliente

id_cliente	nome_cliente	cidade_cliente	id_vendedor
1	Rivaldo	São Paulo	3
2	José	São Paulo	2
3	Ronaldo	São Paulo	1
4	Adalberto	Rio de Janeiro	2
5	Laura	São Paulo	4
6	Elaine	Maranhão	6
7	Fabiano	São Paulo	7
8	Jeff	Curitiba	1
9	Alberto	São Paulo	5
10	Romualdo	São Paulo	1

Vendedores

id_vendedor	nome_vendedor	cidade_vendedor
1	Joao	São Paulo
2	Gilberto	Rio de Janeiro
3	Sara	São Paulo
4	Renata	Curitiba
5	Jaqueline	João Pessoa
6	Reinaldo	Maranhão

Como escrever uma query para trazer *apenas o nome dos vendedores e a sua respectiva lista de clientes?*

Resposta:

```
select
    nome_vendedor,
    nome_cliente
from cliente as t1
left join vendedores as t2
on t1.id vendedor = t2.id vendedor
```



Podemos selecionar quais colunas vão aparecer na tabela resposta ao escolher as colunas na query.
Se usássemos o * traríamos todas as colunas de ambas as tabelas

Criar uma tabela que junta meninos e brinquedos,
apenas com os brinquedos que cada menino possui

Tabela Meninos

menino_id	menino	brinquedo_id
1	Davey	3
2	Bobby	5
3	Beaver	2
4	Richie	1
6	Johnny	4
5	Billy	2

Tabela Brinquedos

id	brinquedo
1	bambole
2	barquinho
3	soldadinho
4	gaita
5	cartoes pokemon
6	lego
7	tectoy
8	sanfona

Como juntar a tabela meninos com a tabela brinquedos?

Criar uma tabela que junta meninos e brinquedos,
apenas com os brinquedos que cada menino possui

Resposta:

```
SELECT
    T1.MENINO,
    T2.BRINQUEDO
FROM MENINOS AS T1
LEFT JOIN BRINQUEDOS AS T2
ON T1.BRINQUEDO_ID = T2.ID;
```

MENINO	BRINQUEDO
Davey	soldadinho
Bobby	cartoes pokemon
Beaver	barquinho
Richie	bambole
Johnny	gaita
Billy	barquinho

Escrever um INNER JOIN de cabeça

Escrever um INNER JOIN de cabeça

Resposta:

```
SELECT *
FROM TABELA1
INNER JOIN TABELA2
ON TABELA1.COL1 = TABELA2.COL2
```

Como aparece a informação na tabela resultado de um LEFT JOIN quando temos valor na tabela da esquerda (Tabela do FROM) mas não temos valor na tabela da direita (Tabela após o JOIN)?

Como aparece a informação na tabela resultado de um LEFT JOIN quando temos valor na tabela da esquerda (Tabela do FROM) mas não temos valor na tabela da direita (Tabela após o JOIN)?

Resposta:

As informações que não estiverem na segunda tabela serão nulas

É possível juntar mais de uma tabela numa única query?

É possível juntar mais de uma tabela numa única query?

Resposta:

Sim.

Explique com as suas palavras a diferença do LEFT JOIN para o INNER JOIN

Explique com suas palavras a diferença do LEFT JOIN para o INNER JOIN

Resposta:

INNER JOIN traz apenas as linhas em comum. LEFT JOIN traz todas as linhas da primeira tabela e se não tiver na segunda tabela traz NULL

Quando faço um LEFT JOIN de uma tabela A com 10 linhas com uma tabela B com 5 linhas, quantas linhas temos como o resultado do LEFT JOIN?

Quando faço um LEFT JOIN de uma tabela A com 10 linhas com uma tabela B com 5 linhas, quantas linhas temos como o resultado do LEFT JOIN?

Resposta:

O número de linhas na tabela de resultado é o total de linhas em A, que é 10.

Quando faço um LEFT JOIN (usando *) de uma tabela A com 3 colunas com uma tabela B com 2 colunas, quantas colunas temos como o resultado do LEFT JOIN?

Quando faço um LEFT JOIN (usando *) de uma tabela A com 3 colunas com uma tabela B com 2 colunas, quantas colunas temos como o resultado do LEFT JOIN?

Resposta:

O número de colunas na tabela de resultado é a soma das colunas em A e B, que é $3 + 2 = 5$.

Descrever as informações que cada uma das tabelas abaixo possui. E se fosse para juntar as duas tabelas, qual seria a coluna de junção?

Produtos

id_produto	nome	preco
1	Caneta Azul	1.50
2	Caderno 100 folhas	10.00
3	Lápis HB	0.50

Pedidos

id_pedido	id_produto	quantidade
1	1	10
2	1	5
3	2	2
4	2	1
5	2	4
6	3	20
7	3	10

Escreva uma consulta SQL que retorne o nome do produto, a quantidade total vendida e a receita total gerada por cada produto.

Tabela Resposta

nome	quantidade_total	receita_total
Caneta Azul	15	22.50
Caderno 100 folhas	7	70.00
Lápis HB	30	15.00

Produtos

id_produto	nome	preco
1	Caneta Azul	1.50
2	Caderno 100 folhas	10.00
3	Lápis HB	0.50

Pedidos

id_pedido	id_produto	quantidade
1	1	10
2	1	5
3	2	2
4	2	1
5	2	4
6	3	20
7	3	10

Escreva uma consulta SQL que retorne o nome do produto, a quantidade total vendida e a receita total gerada por cada produto.

Tabela Resposta

nome	quantidade_total	receita_total
Caneta Azul	15	22.50
Caderno 100 folhas	7	70.00
Lápis HB	30	15.00

Resposta

```
SELECT
    produtos.nome,
    SUM(pedidos.quantidade) as quantidade_total,
    SUM(produtos.preco * pedidos.quantidade) as receita_total
FROM produtos
INNER JOIN pedidos
ON produtos.id_produto = pedidos.id_produto
GROUP BY 1
```

Descrever as informações que cada uma das tabelas abaixo possui. E se fosse para juntar as duas tabelas, qual seria a coluna de junção?

Alunos

id_aluno	nome	curso
1	Ana	Engenharia
2	Bruno	Direito
3	Carla	Medicina
4	Daniel	Engenharia
5	Elisa	Direito
6	Felipe	Medicina
7	Gabriela	Engenharia

Notas

id_aluno	valor
1	7.5
1	8.0
1	6.5
2	9.0
2	6.0
2	7.5
3	8.0
3	8.5
4	7.0
4	8.5
4	9.0
5	6.5
5	7.0
6	8.0
6	7.5
7	9.0
7	9.5

Escreva uma consulta SQL que retorne o nome do aluno, o nome do curso que ele está fazendo e a média das notas desse aluno naquele curso

Tabela Resposta

nome	curso	media_notas
Ana	Engenharia	7.33
Bruno	Direito	7.50
Carla	Medicina	8.25
Daniel	Engenharia	8.17
Elisa	Direito	6.75
Felipe	Medicina	7.75
Gabriela	Engenharia	9.25

Alunos

id_aluno	nome	curso
1	Ana	Engenharia
2	Bruno	Direito
3	Carla	Medicina
4	Daniel	Engenharia
5	Elisa	Direito
6	Felipe	Medicina
7	Gabriela	Engenharia

Notas – Falta id_materia

id_aluno	valor
1	7.5
1	8.0
1	6.5
2	9.0
2	6.0
2	7.5
3	8.0
3	8.5
4	7.0
4	8.5
4	9.0
5	6.5
5	7.0
6	8.0
6	7.5
7	9.0
7	9.5

Escreva uma consulta SQL que retorne o nome do aluno, o nome do curso que ele está fazendo e a média das notas desse aluno.

Tabela Resposta

nome	curso	media_notas
Ana	Engenharia	7.33
Bruno	Direito	7.50
Carla	Medicina	8.25
Daniel	Engenharia	8.17
Elisa	Direito	6.75
Felipe	Medicina	7.75
Gabriela	Engenharia	9.25

Resposta

```
SELECT
    alunos.nome,
    alunos.curso,
    AVG(notas.valor) as media
FROM alunos
INNER JOIN notas
ON alunos.id_aluno = notas.id_aluno
GROUP BY 1,2
```

Desafio: Como encontrar pares de fornecedores que forneçam exatamente as mesmas peças utilizando uma tabela chamada SupParts que contém informações sobre o número do fornecedor (sno) e o número da peça (pno)?

subparts

sno	pno
A	P1
A	P2
A	P3
B	P1
B	P2
C	P1
C	P2
C	P3
D	P1
D	P2

Resposta Desafio: Como encontrar pares de fornecedores que forneçam exatamente as mesmas peças utilizando uma tabela chamada SupParts que contém informações sobre o número do fornecedor (sno) e o número da peça (pno)?

```
WITH supplier_parts AS (
    SELECT sno, array_agg(pno) as parts
    FROM SupParts
    GROUP BY sno
)

SELECT sp1.sno, sp2.sno
FROM supplier_parts sp1
JOIN supplier_parts sp2
ON sp1.sno < sp2.sno
AND sp1.parts = sp2.parts
```

UNION



Existem duas formas de juntar tabelas:

Usando o **JOIN** e usando o **UNION**



Para juntar duas ou mais tabelas na horizontal usamos o JOIN

Suponha que temos duas tabelas: a tabela 1 e a tabela2

Tabela 1

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Tabela 2

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Podemos juntá-las na horizontal...

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100



Para juntar as tabelas na horizontal usamos o JOIN

Para juntar duas ou mais tabelas na vertical usamos o UNION

Tabela 1

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Tabela 2

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

Ou podemos juntá-las na vertical...

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100

ID	A	B	C
1	100	100	100
2	100	100	100
3	100	100	100
4	100	100	100
5	100	100	100



Para juntar as tabelas na vertical usamos o UNION



Precisamos de um relatório com os dados de todos os nossos funcionários mas também dados de nossos clientes.

UNION: como juntar verticalmente duas tabelas?

Vamos analisar as tabelas abaixo:

Funcionários

id	nome	sobrenome	idade
1	Tomas	Miller	22
2	Jean	Silva	26
3	Charles	Madureira	30
4	Lisa	Ferreira	21
5	Ana	Andrade	22
6	James	Smith	20

Cientes

id	nome	sobrenome	idade
1	Ana	Souza	45
2	Mark	Ferreira	21
3	Charles	Madureira	30

Elas trazem:

- O id da pessoa
- O nome
- O sobrenome
- A idade

Como colocar uma tabela embaixo da outra?

UNION ALL

```
SELECT nome, sobrenome, idade FROM funcionarios
UNION ALL
SELECT nome, sobrenome, idade FROM clientes
```

nome	sobrenome	idade
Tomas	Miller	22
Jean	Silva	26
Charles	Madureira	30
Lisa	Ferreira	21
Ana	Andrade	22
James	Smith	20
Ana	Souza	45
Mark	Ferreira	21
Charles	Madureira	30

UNION

```
SELECT nome, sobrenome, idade FROM funcionarios
UNION
SELECT nome, sobrenome, idade FROM clientes
```

nome	sobrenome	idade
Tomas	Miller	22
Jean	Silva	26
Charles	Madureira	30
Lisa	Ferreira	21
Ana	Andrade	22
James	Smith	20
Ana	Souza	45
Mark	Ferreira	21

UNION ALL versus UNION

```
SELECT nome, sobrenome, idade FROM funcionarios
UNION ALL
SELECT nome, sobrenome, idade FROM clientes
```

```
SELECT nome, sobrenome, idade FROM funcionarios
UNION
SELECT nome, sobrenome, idade FROM clientes
```

nome	sobrenome	idade
Tomas	Miller	22
Jean	Silva	26
Charles	Madureira	30
Lisa	Ferreira	21
Ana	Andrade	22
James	Smith	20
Ana	Souza	45
Mark	Ferreira	21
Charles	Madureira	30

nome	sobrenome	idade
Tomas	Miller	22
Jean	Silva	26
Charles	Madureira	30
Lisa	Ferreira	21
Ana	Andrade	22
James	Smith	20
Ana	Souza	45
Mark	Ferreira	21

Repare como o **UNION ALL** mantém todas as linhas mesmo que repetidas, enquanto o **UNION** elimina as linhas repetidas;



UNION ALL repete as linhas
repetidas.
UNION não repete.



Condições para o UNION funcionar:

Os tipos e as quantidades de colunas devem ser as mesmas.

Se os tipos não forem os mesmos o banco de dados fará ‘variable coercion’ e corre o risco de ‘dar ruim’.

Como fazer uma query para trazer uma lista com todos os nomes, independente de ser um nome de vendedor ou de cliente?

Vendedores

id_vendedor	nome_vendedor	cidade_vendedor
1	Joao	São Paulo
2	Gilberto	Rio de Janeiro
3	Sara	São Paulo
4	Renata	Curitiba
5	Jaqueleine	João Pessoa
6	Reinaldo	Maranhão

Cliente

id_cliente	nome_cliente	cidade_cliente	id_vendedor
1	Rivaldo	São Paulo	3
2	José	São Paulo	2
3	Ronaldo	São Paulo	1
4	Adalberto	Rio de Janeiro	2
5	Laura	São Paulo	4
6	Elaine	Maranhão	6
7	Fabiano	São Paulo	7
8	Jeff	Curitiba	1
9	Alberto	São Paulo	5
10	Romualdo	São Paulo	1

TABELA DESEJADA

Adalberto
Alberto
Elaine
Fabiano
Gilberto
Jaqueleine
Jeff
Joao
José
Laura
Reinaldo
Renata
Rivaldo
Romualdo
Ronaldo
Sara

Como fazer uma query para trazer uma lista com todos os nomes, independente de ser um nome de vendedor ou de cliente?

Resposta:

```
select nome_vendedor from vendedores  
union  
select nome_cliente from cliente;
```

nome_vendedor
Adalberto
Alberto
Elaine
Fabiano
Gilberto
Jaqueline
Jeff
Joao
José
Laura
Reinaldo
Renata
Rivaldo
Romualdo
Ronaldo
Sara

Como fazer uma query para trazer uma lista com o nome de todas as cidades? Independente de ser uma cidade de vendedor ou de cliente? Mas esta lista não pode repetir os nomes das cidades

Vendedores

<code>id_vendedor</code>	<code>nome_vendedor</code>	<code>cidade_vendedor</code>
1	Joao	São Paulo
2	Gilberto	Rio de Janeiro
3	Sara	São Paulo
4	Renata	Curitiba
5	Jaqueleine	João Pessoa
6	Reinaldo	Maranhão

Cliente

<code>id_cliente</code>	<code>nome_cliente</code>	<code>cidade_cliente</code>	<code>id_vendedor</code>
1	Rivaldo	São Paulo	3
2	José	São Paulo	2
3	Ronaldo	São Paulo	1
4	Adalberto	Rio de Janeiro	2
5	Laura	São Paulo	4
6	Elaine	Maranhão	6
7	Fabiano	São Paulo	7
8	Jeff	Curitiba	1
9	Alberto	São Paulo	5
10	Romualdo	São Paulo	1

TABELA DESEJADA

Curitiba
João Pessoa
Maranhão
Rio de Janeiro
São Paulo

Como fazer uma query para trazer uma lista com o nome de todas as cidades? Independente de ser uma cidade de vendedor ou de cliente? Mas esta lista não pode repetir os nomes das cidades

Resposta:

```
select cidade_vendedor from vendedores  
union  
select cidade_cliente from cliente;
```

cidade_vendedor
Curitiba
João Pessoa
Maranhão
Rio de Janeiro
São Paulo

Como fazer uma query para trazer uma lista com todos os nomes, e disciplinas independente de ser um nome de aluno ou de professor?

Alunos

nome	disciplina
Maria	Matemática
João	História
Pedro	Geografia
Carol	Matemática
Rafael	Química

Professores

nome	disciplina
Luíza	Química
André	Matemática
Ana	Geografia
Joana	História
Marcos	Física

TABELA DESEJADA

nome	disciplina
Maria	Matemática
João	História
Pedro	Geografia
Carol	Matemática
Rafael	Química
Luíza	Química
André	Matemática
Ana	Geografia
Joana	História
Marcos	Física

Como fazer uma query para trazer uma lista com todos os nomes, e disciplinas independente de ser um nome de aluno ou de professor?

Resposta:

```
SELECT nome, disciplina
FROM alunos
UNION ALL
SELECT nome, disciplina
FROM professores;
```

TABELA DESEJADA

nome	disciplina
Maria	Matemática
João	História
Pedro	Geografia
Carol	Matemática
Rafael	Química
Luíza	Química
André	Matemática
Ana	Geografia
Joana	História
Marcos	Física

Como fazer uma query para trazer uma lista com todos os nomes, e preços independente de ser um produto nacional ou importado?

produtos_nacionais

id	nome	preco
1	Arroz	10.50
2	Feijão	8.90
3	Leite	4.20
4	Café	7.80

produtos_importados

id	nome	preco
1	Vinho Chileno	65.00
2	Queijo Francês	45.90
3	Azeite Italiano	29.99
4	Chocolate Suíço	12.50

TABELA DESEJADA

nome	preco
Arroz	10.50
Feijão	8.90
Leite	4.20
Café	7.80
Vinho Chileno	65.00
Queijo Francês	45.90
Azeite Italiano	29.99
Chocolate Suíço	12.50

Como fazer uma query para trazer uma lista com todos os nomes, e preços independente de ser um produto nacional ou importado?

Resposta:

```
SELECT nome, preco FROM produtos_nacionais  
UNION ALL  
SELECT nome, preco FROM produtos_importados;
```

TABELA DESEJADA

nome	preco
Arroz	10.50
Feijão	8.90
Leite	4.20
Café	7.80
Vinho Chileno	65.00
Queijo Francês	45.90
Azeite Italiano	29.99
Chocolate Suíço	12.50

Qual é a diferença entre o UNION e o JOIN?

Qual é a diferença entre o UNION e o JOIN?

Resposta:

UNION = Junta tabelas na vertical

JOIN = Junta tabelas na horizontal

Qual é a diferença entre o UNION e o UNION ALL?

Qual é a diferença entre o UNION e o UNION ALL?

Resposta:

UNION = Não repete linhas duplicadas

UNION ALL = Repete linhas duplicadas

Desafio: Festa da Firma

Para ajudar a aumentar a conscientização de marketing para o novo Modelo Chi, a equipe de marketing da sua empresa gostaria de realizar uma festa para alguns dos clientes mais ricos da ZoomZoom em Los Angeles, CA.

Para facilitar a festa, eles gostariam que você criasse uma lista de convidados com clientes da ZoomZoom que moram em Los Angeles, CA, bem como vendedores que trabalham na concessionária da ZoomZoom em Los Angeles, CA.

A lista de convidados deve incluir o primeiro nome, o sobrenome e se o convidado é um cliente ou um funcionário.

Desafio: Festa da Firma

Escreva uma consulta que faça uma lista dos clientes da ZoomZoom e funcionários da empresa que moram em Los Angeles, CA. A lista de convidados deve conter o primeiro nome, o sobrenome e se o convidado é um cliente ou um funcionário.

Tabela Clientes

customer_id	first_name	last_name	city	state
1001	John	Smith	Los Angeles	CA
1002	Emma	Johnson	San Diego	CA
1003	Michael	Brown	Los Angeles	CA
1004	Olivia	Davis	San Francisco	CA
1005	Sophia	Garcia	Los Angeles	CA

Tabela Funcionários

employee_id	first_name	last_name	city	state
2001	David	Anderson	Los Angeles	CA
2002	Sarah	Wilson	San Diego	CA
2003	Robert	Martinez	Los Angeles	CA
2004	Emily	Thompson	San Francisco	CA
2005	Daniel	Rodriguez	Los Angeles	CA

Resposta Desafio: Festa da Firma

```
SELECT
    first_name,
    last_name,
    'Customer' AS guest_type
FROM customers
WHERE city = 'Los Angeles' AND state = 'CA'
UNION
SELECT
    first_name,
    last_name,
    'Employee' AS guest_type
FROM employees
WHERE city = 'Los Angeles' AND state = 'CA';
```

Tabela Resposta

first_name	last_name	guest_type
John	Smith	Customer
Michael	Brown	Customer
David	Anderson	Employee
Robert	Martinez	Employee

JOIN COM MÚLTIPLAS TABELAS

```
join_db2.ALUNO  
join_db2.CURSO  
join_db2.ALUNO_CURSO
```

Juntando mais de duas tabelas numa única query

Vamos ver o exemplo da tabela de alunos, cursos e professores:

Aluno

id_aluno	nome
1	Alan
2	Laura
3	Alan
4	Lorena
5	Nicole
6	Tomas

Aluno_Curso

id_aluno	id_curso
1	2
1	3
2	1
2	2
2	3
3	1

Curso

id_curso	nome
1	Bancos de Dados
2	Literatura
3	Programação Python

Como escrever uma query para trazer:
O primeiro nome e o nome dos cursos que cada aluno está
fazendo?

Tabela Desejada:

id_aluno	nome	nome
1	Alan	Literatura
1	Alan	Programação Python
2	Laura	Bancos de Dados
2	Laura	Literatura
2	Laura	Programação Python
3	Alan	Bancos de Dados

Resolução Passo a Passo

Passo1: Juntar as tabelas alunos e aluno_curso

```
SELECT
  *
FROM alunos
INNER JOIN aluno_curso
ON alunos.id_aluno = aluno_curso.id_aluno
```

Resolução Passo a Passo

Passo2: Juntar todo o resultado anterior com a tabela cursos

```
SELECT
  *
FROM alunos
INNER JOIN aluno_curso
ON alunos.id_aluno = aluno_curso.id_aluno
INNER JOIN cursos
ON aluno_curso.id_curso = cursos.id_curso
```

Resolução Passo a Passo

Passo3: Escolher apenas as colunas desejadas para serem exibidas como resposta

```
SELECT
    alunos.id_aluno,
    alunos.nome,
    cursos.nome
FROM alunos
INNER JOIN aluno_curso
ON alunos.id_aluno = aluno_curso.id_aluno
INNER JOIN cursos
ON aluno_curso.id_curso = cursos.id_curso
```

join_db2.ALUNO
join_db2.CURSO
join_db2.ALUNO_CURSO

O que acontece nos bastidores?

Passo1: Juntar as tabelas alunos e aluno_curso

id_aluno	nome	id_aluno	id_curso
1	Alan	1	2
1	Alan	1	3
2	Laura	2	1
2	Laura	2	2
2	Laura	2	3
3	Alan	3	1

O que acontece nos bastidores?

Passo2: Juntar todo o resultado anterior com a tabela cursos

id_aluno	nome	id_aluno	id_curso	nome
1	Alan		1	2 Literatura
1	Alan		1	3 Programação Python
2	Laura		2	1 Bancos de Dados
2	Laura		2	2 Literatura
2	Laura		2	3 Programação Python
3	Alan		3	1 Bancos de Dados

O que acontece nos bastidores?

Passo3: Escolher apenas as colunas desejadas para serem exibidas como resposta

id_aluno	nome	nome
1	Alan	Literatura
1	Alan	Programação Python
2	Laura	Bancos de Dados
2	Laura	Literatura
2	Laura	Programação Python
3	Alan	Bancos de Dados

join_db2.ALUNO
join_db2.CURSO
join_db2.ALUNO_CURSO

A query final já faz os três passos anteriores e mostra apenas a tabela resultado

```
SELECT
    alunos.id_aluno,
    alunos.nome,
    cursos.nome
FROM alunos
INNER JOIN aluno_curso
ON alunos.id_aluno = aluno_curso.id_aluno
INNER JOIN cursos
ON aluno_curso.id_curso = cursos.id_curso
```

id_aluno	nome	nome
1	Alan	Literatura
1	Alan	Programação Python
2	Laura	Bancos de Dados
2	Laura	Literatura
2	Laura	Programação Python
3	Alan	Bancos de Dados



A ordem de execução do JOIN nos bastidores é:

1. Junta uma tabela com a outra e forma um tabelão
2. Filtra
3. Exibe as colunas que você quer



A primeira tabela que se escreve no JOIN é a mais importante, pois ela define a estrutura da tabela resultado.

Como juntar as tabelas clientes, produtos e pedidos para gerar a tabela abaixo?

Clientes

id	nome	email
1	João	joao@example.com
2	Maria	maria@example.com
3	Ana	ana@example.com
4	Luiz	luiz@example.com

Produtos

id	nome	preço
1	Camiseta	29.99
2	Calça jeans	79.99
3	Tênis	99.99
4	Jaqueta	149.99

Pedidos

id	id_cliente	id_produto	quantidade
1	1	1	2
2	1	2	1
3	2	1	3
4	2	3	2

TABELA DESEJADA

nome do cliente	nome do produto
João	Camiseta
João	Calça jeans
Maria	Camiseta
Maria	Tênis
Ana	Tênis
Ana	Jaqueta

Como juntar as tabelas clientes, produtos e pedidos para gerar a tabela abaixo?

Resposta:

```
SELECT
    clientes.nome,
    produtos.nome
FROM pedidos
JOIN clientes ON clientes.id = pedidos.id_cliente
JOIN produtos ON produtos.id = pedidos.id_produto;
```

Criar uma tabela que traga o estoque de cada brinquedo em cada loja

LOJA_INFO

loja_id	endereco	tel	gerente
1	23 Maple	555-6712	Joe
2	1902 Amber Ln	555-3478	Susan
3	100 E. North St.	555-0987	Tara
4	17 Engleside	555-6554	Gordon

TABELA DESEJADA

LOJA_ID	GERENTE	COALESCE(BRINQUEDO, 'SEM REGISTRO')	ESTOQUES
1	Joe	cartoes pokemon	5
1	Joe	cartoes pokemon	34
1	Joe	SEM REGISTRO	50
2	Susan	lego	10
2	Susan	SEM REGISTRO	2
2	Susan	SEM REGISTRO	18
3	Tara	cartoes pokemon	12
4	Gordon	lego	24
4	Gordon	SEM REGISTRO	11
4	Gordon	SEM REGISTRO	28

ESTOQUES

brinquedo_id	loja_id	estoques
5	1	34
5	3	12
5	1	5
6	2	10
6	4	24
9	1	50
9	2	2
9	2	18
12	4	28
12	4	11

BRINQUEDOS

id	brinquedo
1	bambole
2	barquinho
3	soldadinho
4	gaita
5	cartoes pokemon
6	lego
7	tectoy
8	sanfona

Resolução

```
SELECT
    LOJA_ID,
    GERENTE,
    COALESCE(BRINQUEDO, 'SEM REGISTRO'),
    ESTOQUES
FROM LOJA_INFO AS T1
LEFT JOIN ESTOQUES AS T2
ON T1.LOJA_ID = T2.LOJA_ID
LEFT JOIN BRINQUEDOS AS T3
ON T2.BRINQUEDO_ID = T3.ID;
```

LOJA_ID	GERENTE	COALESCE(BRINQUEDO, 'SEM REGISTRO')	ESTOQUES
1	Joe	cartoes pokemon	5
1	Joe	cartoes pokemon	34
1	Joe	SEM REGISTRO	50
2	Susan	lego	10
2	Susan	SEM REGISTRO	2
2	Susan	SEM REGISTRO	18
3	Tara	cartoes pokemon	12
4	Gordon	lego	24
4	Gordon	SEM REGISTRO	11
4	Gordon	SEM REGISTRO	28

Escrever uma query que junta as três tabelas abaixo e traz apenas as colunas de nome do cliente e produto comprado

Clientes

id	nome	email	telefone
1	Ana	ana@email.com	(11) 9999-8888
2	João	joao@email.com	(21) 5555-4444
3	Maria	maria@email.com	(31) 3333-2222

Produtos

id	nome	preco
1	Smartphone	1500.00
2	Fone de ouvido	200.00
3	Notebook	2800.00
4	Tênis	100.00
5	Camiseta	50.00

Pedidos

id	id_cliente	id_produto	quantidade	data
1	1	1	1	02/01/2023
2	1	2	2	03/01/2023
3	2	4	3	05/01/2023
4	2	5	2	01/01/2023

TABELA DESEJADA

nome	produto_comprado
Ana	Smartphone
Ana	Fone de ouvido
João	Tênis
João	Camiseta
Maria	Notebook
Maria	Camiseta
Maria	Camiseta

Escrever uma query que junta as três tabelas abaixo e traz apenas as colunas de nome do cliente e produto comprado

Resposta:

```
SELECT
    clientes.nome,
    produtos.nome AS produto_comprado
FROM clientes
JOIN pedidos ON clientes.id = pedidos.id_cliente
JOIN produtos ON produtos.id = pedidos.id_produto;
```

Desafio: Como produzir um relatório com informações sobre os pagamentos de aluguel em um complexo de apartamentos?

units

unit_nbr	complex_id	area
1	32	100
2	32	150
3	32	120
4	32	90
5	32	110
6	32	130
7	33	---
8	33	
9	33	
10	33	

tenants

tenant_id	tenant_name	unit_nbr	vacated_date
1	John Smith	1	
2	Jane Doe	2	
3	Bob Johnson	3	
4	Michael Lee	4	
5	Emily Davis	5	
6	David Wilson	6	
7	Daniel Brown	7	
8	Matthew Taylor	8	
9	Ryan Harris	9	
10	Joseph Garcia	10	

rentpayments

tenant_id	unit_nbr	payment_date	amount
1	1	2021-01-01	800
2	2	2021-01-02	750
3	3	2021-01-03	700
4	4	2021-01-04	650
5	5	2021-01-05	600
6	6	2021-01-06	550
7	7	2021-01-07	500
8	8	2021-01-08	450
9	9	2021-01-09	400
10	10	2021-01-10	350

Resposta Desafio: Como produzir um relatório com informações sobre os pagamentos de aluguel em um complexo de apartamentos?

```
SELECT
*
FROM RENTPAYMENTS AS RP1
INNER JOIN UNITS AS U1
ON RP1.UNIT_BR = U1.UNIT_BR
INNER JOIN TENANTS AS T1
ON T1.TENANT_ID = RP1.TENANT_ID
```

Desafio: Prospect de Remarketing

O objetivo desta atividade é limpar e preparar nossos dados para análise usando técnicas SQL.

A equipe de ciência de dados deseja construir um novo modelo para ajudar a prever quais clientes são os melhores prospects para remarketing.

Um novo cientista de dados se juntou à equipe e não conhece bem o banco de dados o suficiente para extrair um conjunto de dados para este novo modelo.

A responsabilidade agora recai sobre você para ajudar o novo cientista de dados a preparar e construir um conjunto de dados a ser usado para treinar um modelo.

Escreva consultas ou queries para montar um conjunto de dados que faça o seguinte:

Desafio: Prospect de Remarketing

1. Use INNER JOIN para unir a tabela de clientes à tabela de vendas.
2. Use INNER JOIN para unir a tabela de produtos à tabela de vendas.
3. Use LEFT JOIN para unir a tabela de concessionárias à tabela de vendas.
4. Agora, retorne todas as colunas da tabela de clientes e da tabela de produtos.
5. Em seguida, retorne a coluna id_concessionaria da tabela de vendas, mas preencha id_concessionaria em vendas com -1 se for NULL.
6. Adicione uma coluna chamada high_savings que retorne 1 se o valor da venda foi 500 menor que o base_msrp ou menos. Caso contrário, retorna 0.

Tabela Vendas

venda_id	cliente_id	produto_id	concessionaria_id	valor	base_msrp
1	1001	2001	3001	1200	1500
2	1002	2002	NULL	800	1000
3	1003	2003	3002	450	500
4	1004	2004	3003	300	350
5	1005	2005	NULL	1800	2000

Tabela Produtos

produto_id	nome
2001	Produto A
2002	Produto B
2003	Produto C
2004	Produto D
2005	Produto E

Tabela Clientes

cliente_id	nome
1001	Cliente 1
1002	Cliente 2
1003	Cliente 3
1004	Cliente 4
1005	Cliente 5

Tabela Concessionárias

concessionaria_id	nome
3001	Concessionária A
3002	Concessionária B
3003	Concessionária C

Resposta Desafio: Lista de Clientes

```
SELECT
    c.*,
    v.*,
    COALESCE(v.concessionaria_id, -1) AS concessionaria_id,
    CASE
        WHEN v.valor <= (v.base_msrp - 500) THEN 1
        ELSE 0
    END AS high_savings
FROM
    clientes AS c
INNER JOIN vendas AS v ON c.cliente_id = v.cliente_id
INNER JOIN produtos p ON p.produto_id = v.produto_id
LEFT JOIN concessionarias AS conc
ON conc.concessionaria_id = s.concessionaria_id;
```

cliente_id	nome	produto_id	nome	concessionaria_id	high_savings
1001	Cliente 1	2001	Produto A	3001	0
1002	Cliente 2	2002	Produto B	-1	1
1003	Cliente 3	2003	Produto C	3002	1
1004	Cliente 4	2004	Produto D	3003	1
1005	Cliente 5	2005	Produto E	-1	0

ANEXOS

SUBQUERY

SQL Avançado: SubQuery

A subquery é uma consulta (query) dentro de outra consulta (query).

Suponha que temos duas tabelas: "Clientes" e "Pedidos". Queremos encontrar todos os clientes que fizeram pedidos nos últimos 30 dias. Podemos usar uma subquery para realizar essa tarefa.

```
SELECT nome  
FROM Clientes  
WHERE id IN (SELECT cliente_id  
              FROM Pedidos  
              WHERE data_pedido >= DATE('now', '-30 days'))
```

Repare que **dentro do WHERE** termos outra query (**SELECT cliente_id FROM pedidos ...**) Os parênteses definem o início e o fim da subquery

PARTIÇÕES

SQL Avançado: Partições – OVER

No SQL, a cláusula OVER é suportada e pode ser usada em conjunto com a função SUM() para realizar cálculos agregados com partições e ordenação. Aqui está um exemplo de uso do SUM() OVER(PARTITION BY ORDER BY) no SQL

```
SELECT
    produto,
    data,
    quantidade,
    SUM(quantidade) OVER(PARTITION BY produto ORDER BY data) AS quantidade_acumulada
FROM vendas;
```

produto	data	quantidade
A	2022-01-01	10
A	2022-01-02	15
B	2022-01-01	5
B	2022-01-02	8

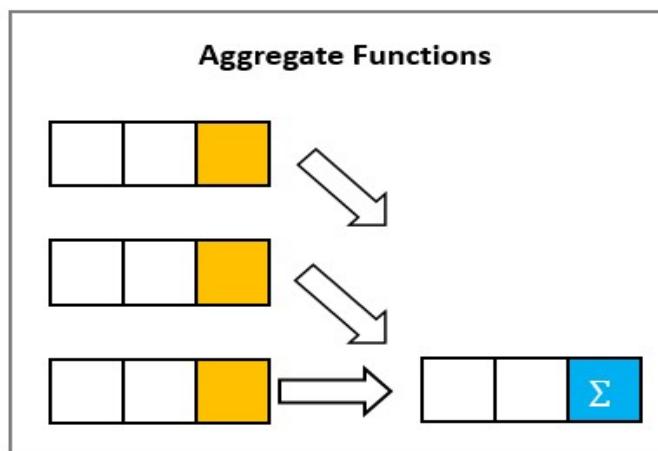


produto	data	quantidade	quantidade_acumulada
A	2022-01-01	10	10
A	2022-01-02	15	25
B	2022-01-01	5	5
B	2022-01-02	8	13

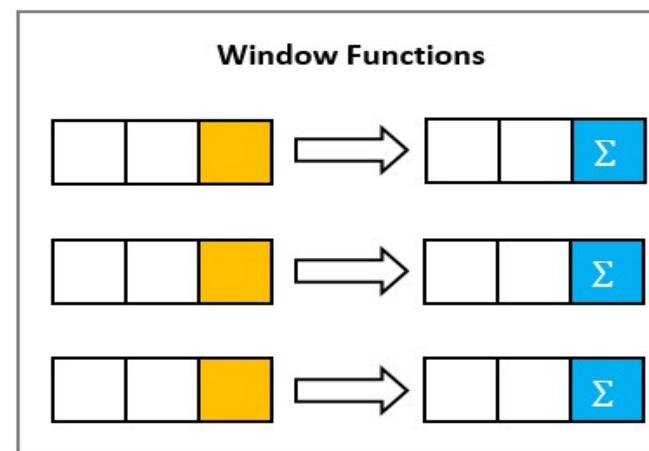


Partições – Por que fazer e como funciona?

Às vezes você quer fazer a soma/média de um grupo e repetir aquela soma para todos os elementos do grupo



No Group By temos a soma do grupo numa nova tabela



No Partition By temos a soma do grupo, com o valor da soma repetido para cada linha

OVER é análogo ao GROUP BY

SELECT

```
ID, NOME, VENDAS, DATA  
SUM(VENDAS) OVER (PARTITION BY NOME)
```

FROM VENDAS

Depois colocamos o ORDER BY dentro do OVER

SELECT

```
ID, NOME, VENDAS, DATA  
SUM(VENDAS) OVER (PARTITION BY NOME ORDER BY DATA)
```

FROM VENDAS

A partição é o grupo que você quer agregar

Racional para pensar em partições

1. Qual coluna será agregada?
2. Qual função de agregação utilizaremos?
3. Qual é a partição? (Pensar como se fosse um group by)
4. Qual é o critério de ordenação?

Partições – Média

No SQL, a cláusula OVER é suportada e pode ser usada em conjunto com a função SUM() para realizar cálculos agregados com partições e ordenação. Aqui está um exemplo de uso do SUM() OVER(PARTITION BY ORDER BY) no SQL

```
SELECT
    produto,
    data,
    quantidade,
    AVG(quantidade) OVER(PARTITION BY produto ORDER BY data) AS quantidade_acumulada
FROM vendas;
```

produto	data	quantidade
A	2022-01-01	10
A	2022-01-02	15
B	2022-01-01	5
B	2022-01-02	8



produto	data	quantidade	media_quantidade
A	2022-01-01	10	10
A	2022-01-02	15	12.5
B	2022-01-01	5	5
B	2022-01-02	8	6.5



Partições – Máximo

No SQL, a cláusula OVER é suportada e pode ser usada em conjunto com a função SUM() para realizar cálculos agregados com partições e ordenação. Aqui está um exemplo de uso do SUM() OVER(PARTITION BY ORDER BY) no SQL

```
SELECT
    produto,
    data,
    quantidade,
    MAX(quantidade) OVER(PARTITION BY produto ORDER BY data) AS quantidade_acumulada
FROM vendas;
```

produto	data	quantidade
A	2022-01-01	10
A	2022-01-02	15
B	2022-01-01	5
B	2022-01-02	8



produto	data	quantidade	max_quantidade
A	2022-01-01	10	10
A	2022-01-02	15	15
B	2022-01-01	5	5
B	2022-01-02	8	8



Funções de ranking

- As funções de ranking criam uma coluna com a ordem de cada linha
- As mais usadas são: RANK e ROW_NUMBER
- Elas também usam a lógica do OVER

SQL Avançado: Partições – RANK

A função RANK() nos permite atribuir uma classificação às linhas de uma consulta com base em uma determinada ordem, permitindo identificar a posição relativa de cada registro em relação aos demais com base no critério de ordenação

```
SELECT
    produto,
    data,
    valor,
    RANK() OVER(ORDER BY valor DESC) AS classificacao
FROM vendas;
```

produto	data	valor
A	2022-01-01	100
B	2022-01-02	150
C	2022-01-03	120
D	2022-01-04	180
E	2022-01-05	200



produto	data	valor	classificacao
E	2022-01-05	200	1
D	2022-01-04	180	2
B	2022-01-02	150	3
C	2022-01-03	120	4
A	2022-01-01	100	5

SQL Avançado: Partições – ROW_NUMBER

O ROW_NUMBER() nos permite atribuir um número sequencial a cada linha, o que pode ser útil para classificar, enumerar ou identificar registros em um conjunto de resultados.

```
SELECT
    nome,
    preco,
    ROW_NUMBER() OVER(ORDER BY preco) AS numero_sequencial
FROM produtos;
```

nome	preco
Produto1	100
Produto2	150
Produto3	120
Produto4	180
Produto5	200

nome	preco	numero_sequencial
Produto1	100	1
Produto3	120	2
Produto2	150	3
Produto4	180	4
Produto5	200	5

SQL Avançado: Diferença entre ROW_NUMBER, RANK e DENSE_RANK

V	ROW_NUMBER	RANK	DENSE_RANK
a	1	1	1
a	2	1	1
a	3	1	1
b	4	4	2
c	5	5	3
c	6	5	3
d	7	7	4
e	8	8	5

Repare que row_number é indiferente em caso de empate

Já o rank atribui o mesmo valor em caso de empate mas o próximo número da sequência dá ‘saltos’

dense_rank atribui o mesmo valor em caso de empate mas o próximo número da sequência não dá ‘saltos’

SQL Avançado: Partições – LAG

O LAG() nos permite acessar o valor de uma coluna em uma linha anterior, o que pode ser útil para realizar cálculos ou comparações com base nos valores anteriores de uma sequência de dados

```
SELECT data, valor, LAG(valor) OVER(ORDER BY data) AS valor_anterior  
FROM vendas;
```

data	valor
2022-01-01	100
2022-01-02	150
2022-01-03	120
2022-01-04	180
2022-01-05	200



data	valor	valor_anterior
2022-01-01	100	null
2022-01-02	150	100
2022-01-03	120	150
2022-01-04	180	120
2022-01-05	200	180

SQL Avançado: Partições – LEAD

O LEAD() nos permite acessar o valor de uma coluna em uma linha seguinte, o que pode ser útil para realizar cálculos ou comparações com base nos valores seguintes de uma sequência de dados.

```
SELECT data, valor, LEAD(valor) OVER(ORDER BY data) AS valor_seguinte  
FROM vendas;
```

data	valor
2022-01-01	100
2022-01-02	150
2022-01-03	120
2022-01-04	180
2022-01-05	200



data	valor	valor_seguinte
2022-01-01	100	150
2022-01-02	150	120
2022-01-03	120	180
2022-01-04	180	200
2022-01-05	200	null

ARRAYS

Arrays em SQL – Por que fazer e como funciona?

Opção1 - Normalizado

name	location
Cafe Paci	Newtown
Ho Jiak	Haymarket
Spice I Am	Surry Hills

name	cuisine_label
Cafe Paci	European
Cafe Paci	Casual
Cafe Paci	Wine bar
Ho Jiak	Malaysian
Ho Jiak	Street-food
Spice I Am	Thai

Opção2 - Tabelão

name	location	cuisine_label
Cafe Paci	Newtown	European
Cafe Paci	Newtown	Casual
Cafe Paci	Newtown	Wine bar
Ho Jiak	Haymarket	Malaysian
Ho Jiak	Haymarket	Street-food
Spice I Am	Thai	Thai

Opção3 – Usando Arrays

name	location	cuisine_label
Cafe Paci	Newtown	European Casual Wine bar
Ho Jiak	Haymarket	Malaysian Street-food
Spice I Am	Surry Hills	Thai

Na imagem acima temos 3 formas diferentes de capturar a relação entre o nome de um restaurante, sua localização e o tipo de cozinha.

Na opção 1 temos 2 tabelas separadas vinculadas por uma relação de um-para-muitos. A normalização ajuda a evitar o armazenamento redundante da localização, reduzindo assim os custos de armazenamento. Mas também estamos sacrificando o desempenho da consulta, pois é necessário fazer uma junção (join) das 2 tabelas se quisermos saber onde fica o restaurante tailandês.

A opção 2 é o completo oposto. O desempenho da consulta será muito mais rápido, pois não é necessário fazer uma junção (join) das 2 tabelas, mas os custos de armazenamento serão maiores, pois a localização é armazenada de forma redundante e a tabela está desnormalizada.

Mas e se pudéssemos ter o melhor dos dois mundos? Com a opção 3, usando arrays – implementada em alguns fornecedores de bancos de dados. Os tipos de cozinha de cada restaurante estão aninhados em uma única linha.

Benefícios de usar Arrays em SQL

1. Melhor performance de queries
2. Menores custos de armazenamento

SQL Avançado: Inserindo dados com Array

- Podemos criar um array com colchetes [] com cada elemento separado por vírgula. Os valores dentro de cada par de colchetes formam 1 array.
- Como cada array representa os rótulos de cozinha associados a um restaurante, o SQL os armazenou em campos repetidos associados a uma única linha. No total temos 4 linhas diferentes correspondentes a 4 restaurantes.

```
INSERT INTO cozinha (nome, localizacao, cozinhas)
VALUES
('Cafe Paci', 'Newtown', ['Europeu', 'Casual', 'Wine bar']),
('Ho Jiak', 'Haymarket', ['Malaio', 'Comida de rua']),
('Spice I Am', 'Surry Hills', ['Tailandês', 'Casual']),
('Chaco Bar', 'Potts Point', ['Japonês', 'Yakitori', 'Casual']);
```

nome	localizacao	cozinhas
Cafe Paci	Newtown	[Europeu, Casual, Wine bar]
Ho Jiak	Haymarket	[Malaio, Comida de rua]
Spice I Am	Surry Hills	[Tailandês, Casual]
Chaco Bar	Potts Point	[Japonês, Yakitori, Casual]

SQL Avançado: Arrays – UNNEST

O UNNEST expande os elementos individuais do array em linhas separadas. Agora, cada valor do array é tratado como um registro separado, permitindo que você realize operações e consultas mais detalhadas nos dados

```
SELECT id, valor  
FROM dados, UNNEST(valores) AS valor
```

id	valores
1	[10, 20, 30]
2	[40, 50]
3	[60, 70, 80]

id	valor
1	10
1	20
1	30
2	40
2	50
3	60
3	70
3	80

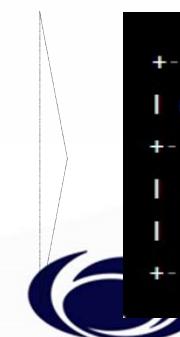


SQL Avançado: Arrays – Filtro com UNNEST

Apenas os restaurantes que possuem a etiqueta "Casual" no array de cozinhas são retornados. O UNNEST é usado para expandir o array "cozinhas" em linhas separadas, e o WHERE IN filtra os resultados para incluir apenas aqueles em que "Casual" está presente no array de cozinhas.

```
SELECT nome, local, cozinha
FROM restaurantes, UNNEST(cozinhas) AS cozinha
WHERE "Casual" IN UNNEST(cozinhas);
```

nome	local	cozinhas
A	Cidade X	[Casual, Fino]
B	Cidade Y	[Rápido, Casual]
C	Cidade Z	[Italiano]



The logo of Faculdade do Grupo Etapa, featuring a stylized blue and white swoosh graphic next to the text "FACULDADE DO GRUPO ETAPA".

nome	local	cozinha
A	Cidade X	Casual
B	Cidade Y	Casual

FACULDADE DO GRUPO ETAPA

SQL Avançado: Arrays – ARRAY_AGG

O ARRAY_AGG agrupa os produtos de cada ID em um único array. Agora, podemos trabalhar com os produtos agrupados como um conjunto, realizar operações em massa ou até mesmo desagregar os arrays em linhas separadas usando funções como UNNEST. O ARRAY_AGG é particularmente útil para agregações em consultas que envolvem agrupamento de dados.

```
SELECT id, ARRAY_AGG(produto) AS produtos  
FROM produtos  
GROUP BY id
```

id	produto	valor
1	A	10
1	B	20
2	C	30
2	D	40
2	E	50

id	produtos
1	[A, B]
2	[C, D, E]



SQL Avançado: Arrays – ARRAY_LENGTH

O ARRAY_LENGTH retorna o comprimento de cada array. O ARRAY_LENGTH é útil quando precisamos realizar operações ou análises com base no tamanho dos arrays em nosso conjunto de dados

```
SELECT id, ARRAY_LENGTH(produtos) AS comprimento  
FROM produtos
```

id	produtos
1	[A, B, C]
2	[D, E]
3	[F, G, H, I]

id	comprimento
1	3
2	2
3	4

ÍNDICES

SQL Avançado: Índices

O índice em SQL é uma estrutura otimizada que acelera as consultas, permitindo uma busca mais rápida e eficiente nos dados da tabela.

Suponha que temos uma tabela chamada "Clientes" com uma coluna "nome" e queremos criar um índice para acelerar as consultas de busca por nome.

```
CREATE INDEX idx_nome ON Clientes(nome);
```

Isso criará um índice chamado "idx_nome" na coluna "nome" da tabela "Clientes". Agora, ao realizar consultas que envolvem a coluna "nome", o SQL usará o índice para acelerar a busca. Por exemplo:

```
SELECT * FROM Clientes WHERE nome = 'João';
```

Com o índice criado, a busca pelo nome 'João' será **mais rápida**, pois o SQL utilizará o índice para localizar rapidamente as linhas correspondentes na tabela.

LINHA DO TEMPO / PERSONAGENS

Linha do Tempo. Eventos em SQL



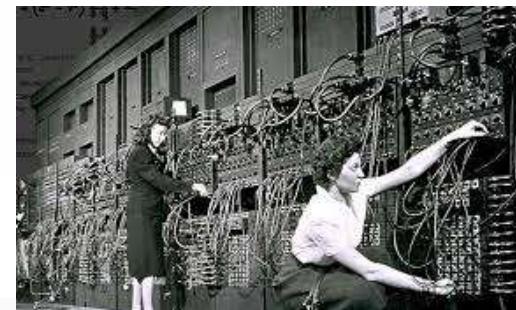
Grandes nomes: Clifford Berry

Clifford Berry (1918-1969) foi um engenheiro americano, co-criador do ENIAC, o primeiro computador programável.



Trabalhou na Moore School of Electrical Engineering. Seu computador, o ENIAC ajudou durante a Segunda Guerra Mundial, pesava 30 toneladas e tinha 17.468 válvulas.

Foi pioneiro na computação e recebeu vários prêmios. Indicado ao Hall da Fama dos Inventores. Era casado e tinha um filho.



Grandes nomes: Edgar Frank Codd

Edgar Frank Codd (1923-2003) foi um cientista da computação britânico, conhecido como o pai do modelo de banco de dados relacional.

Trabalhou na IBM por mais de 40 anos e criou um jeito comum de organizar dados usando tabelas, colunas e linhas.

Seu modelo é amplamente usado hoje e foi premiado com o Prêmio Turing em 1971. Codd foi muito influente na computação do século 20 e mudou a forma como armazenamos e organizamos informações



Information Retrieval

A Relational Model of Data for
Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network

Grandes nomes: Donald Chamberlin e Raymond Boyce

Donald Chamberlin é um cientista da computação americano, famoso por co-criar o SQL, uma linguagem de consulta para bancos de dados.

Formado em matemática em 1954, trabalhou na IBM e desenvolveu o SQL em 1970 com Raymond Boyce.

Chamberlin se aposentou em 1995, mas continuou sua pesquisa e recebeu prêmios por seu trabalho no SQL. Seu trabalho impactou o desenvolvimento de bancos de dados, tornando mais rápido e fácil para as pessoas acessarem e modificarem dados.



Grandes nomes: Chris Date

Chris Date é um cientista da computação britânico especializado em bancos de dados relacionais.

Nasceu em 1940, se formou em matemática na Universidade de Cambridge e trabalhou na IBM, ajudando a desenvolver o modelo relacional.

Fundou a Date Foundation em 1973 e é autor de vários livros sobre bancos de dados. É membro da Academia Nacional de Engenharia e recebeu o Prêmio Turing em 2003.

