

SQL

Módulo MySQL



SUMÁRIO

MÓDULO 1	Introdução	001
<hr/>		
MÓDULO 2	Instalação e Importação do Banco de Dados	029
<hr/>		
MÓDULO 3	Consultas Básicas	033
<hr/>		
MÓDULO 4	Filtros no MySQL	044
<hr/>		
MÓDULO 5	Operações Básicas e Funções de Agregação	055
<hr/>		
MÓDULO 6	Agrupamentos no MySQL	066
<hr/>		
MÓDULO 7	Variáveis no MySQL	076
<hr/>		
MÓDULO 8	Funções de Texto e Data no MySQL	085
<hr/>		
MÓDULO 9	Joins	104
<hr/>		
MÓDULO 10	Funções Condicionais	134
<hr/>		



SUMÁRIO

MÓDULO 11 Views

151

MÓDULO 12 CRUD

170

MÓDULO 13 Functions e Procedures

204

MÓDULO 14 Subqueries

227



MÓDULO 1

INTRODUÇÃO

INTRODUÇÃO

INTRODUÇÃO



O que é um Banco de Dados?

Um banco de dados é um conjunto de dados organizados dentro de uma ou mais tabelas, que terão alguma relação entre si.

Partimos de uma informação isolada, na sua forma mais simples, e a partir de um conjunto de diversas informações isoladas, conseguimos organizar essas informações para criar tabelas e bancos de dados.



O que é um Banco de Dados?

“

Bancos de dados são conjuntos de tabelas, com alguma relação entre si, com dados sobre pessoas, lugares ou coisas.

Estes dados organizados permitem a compreensão de um determinado fenômeno na empresa, seja a preferência dos usuários em uma rede social, seja o perfil de consumo em um aplicativo de transações financeiras.

Do que é composto um banco de dados?

Um banco de dados é composto pelos seguintes elementos:

- ✓ Tabelas
- ✓ Campos (Atributos)
- ✓ Registros (Tuplas)

Do que é composto um banco de dados?

Um banco de dados é composto pelos seguintes elementos:

- Tabelas
- Campos (Atributos)
- Registros (Tuplas)

Tabelas armazenam dados dentro de um banco de dados.

Uma tabela é um conjunto de dados relacionados, e é composta por linhas e colunas.

Do que é composto um banco de dados?

Um banco de dados é composto pelos seguintes elementos:

 Tabelas

 Campos (Atributos)

 Registros (Tuplas)

Campos (também chamados de atributos ou simplesmente colunas) representam os atributos dos dados, como ID do Produto, Nome do Cliente, Quantidade Vendida, Preço, etc.

Um campo é uma coluna de uma tabela, contendo informações específicas sobre cada registro (linha) da tabela.

Do que é composto um banco de dados?

Um banco de dados é composto pelos seguintes elementos:

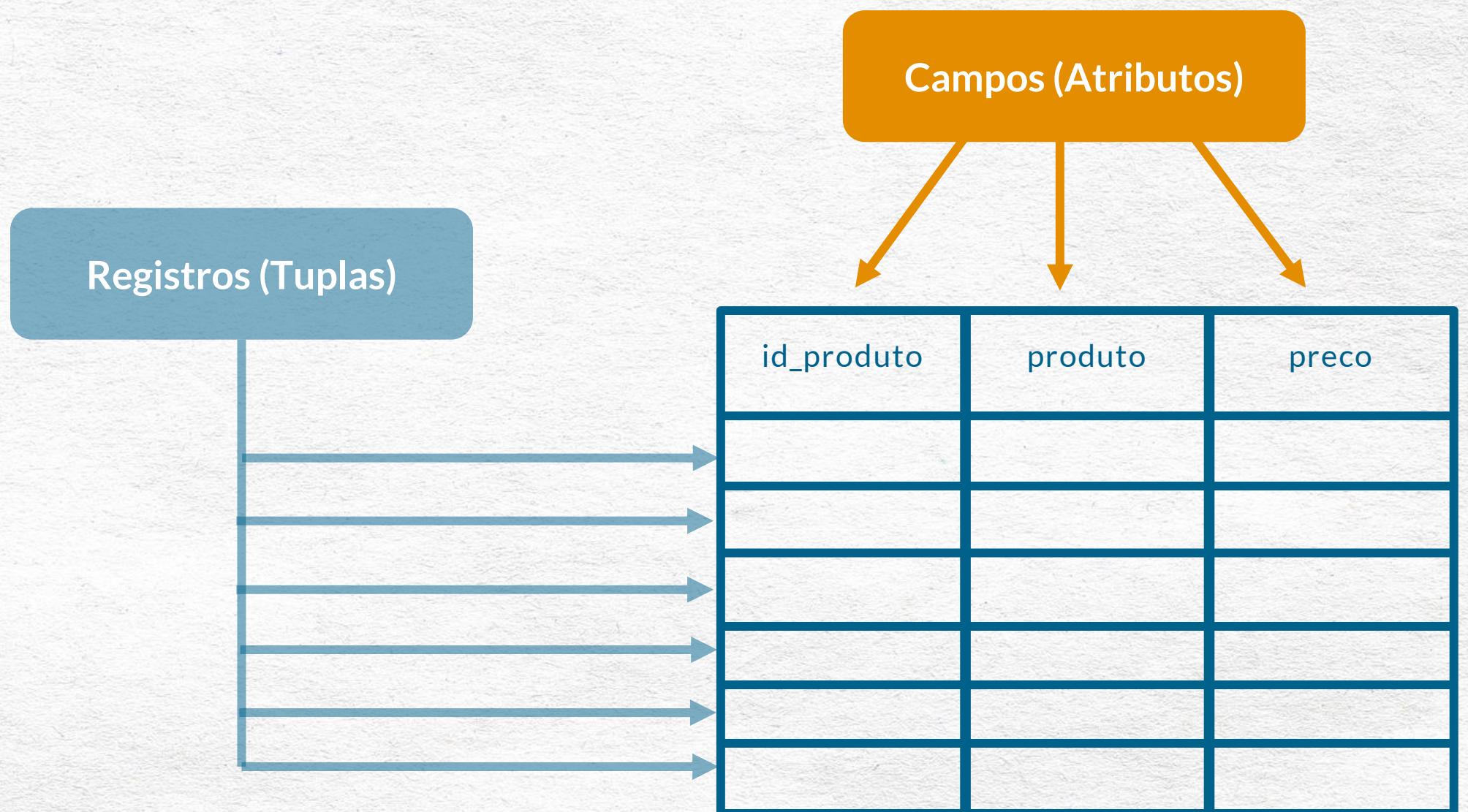
- Tabelas
- Campos (Atributos)
- Registros (Tuplas)

Registros (também chamados de tuplas ou simplesmente linhas) representam cada entrada de dados dentro de uma tabela.

Do que é composto um banco de dados?

Um banco de dados é composto pelos seguintes elementos:

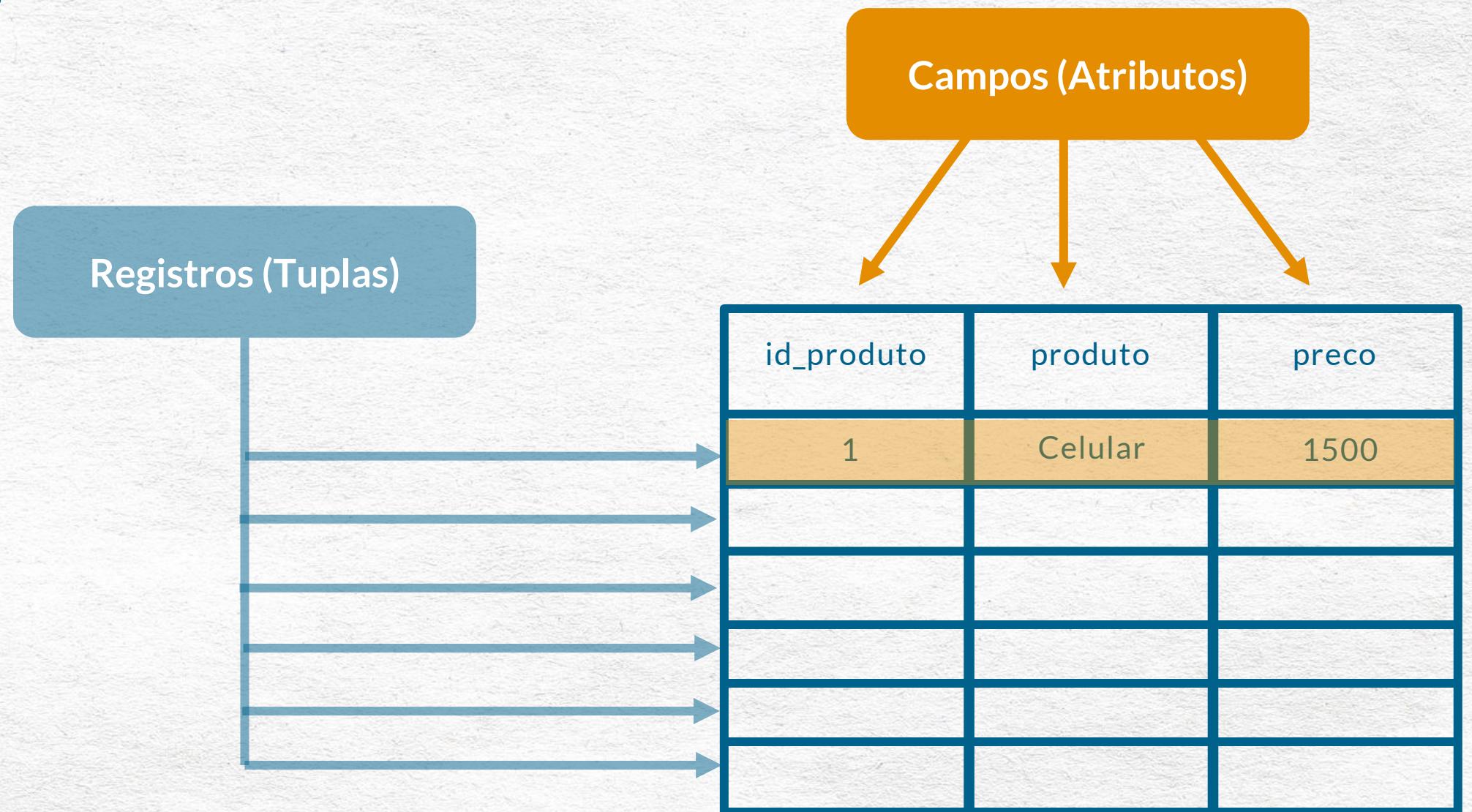
- Tabelas
- Campos (Atributos)
- Registros (Tuplas)



Do que é composto um banco de dados?

Um banco de dados é composto pelos seguintes elementos:

- Tabelas
- Campos (Atributos)
- Registros (Tuplas)

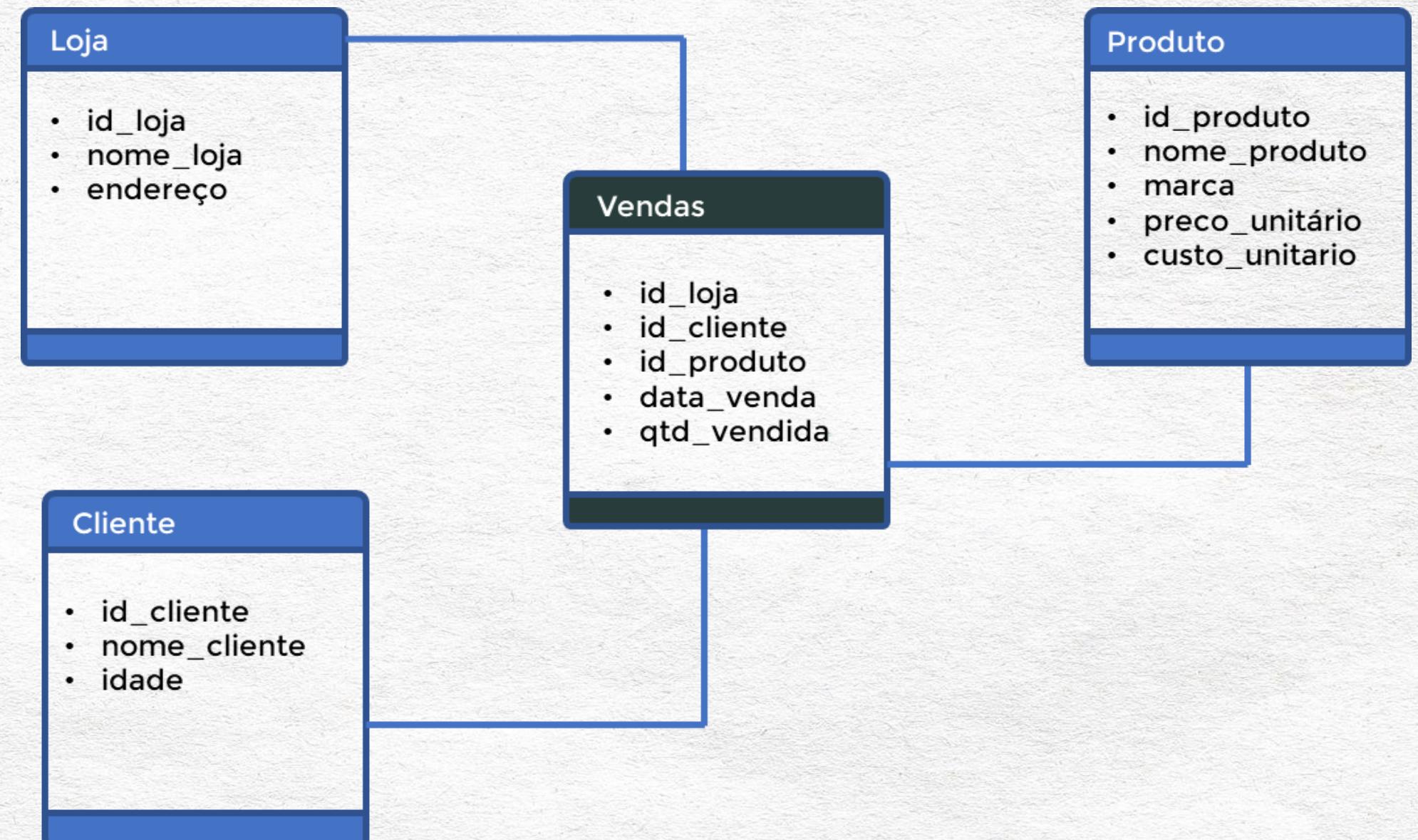


O que é um Banco de Dados?

O desenho esquemático de um banco de dados é mostrado ao lado. Diversas tabelas, com diferentes informações sobre um negócio, e que possuem algum tipo de relação entre si.

A esse banco de dados damos o nome de **RELACIONAL**.

Bancos de dados relacionais serão o foco do nosso curso, até por serem o tipo de bancos de dados mais comumente encontrados no mercado.

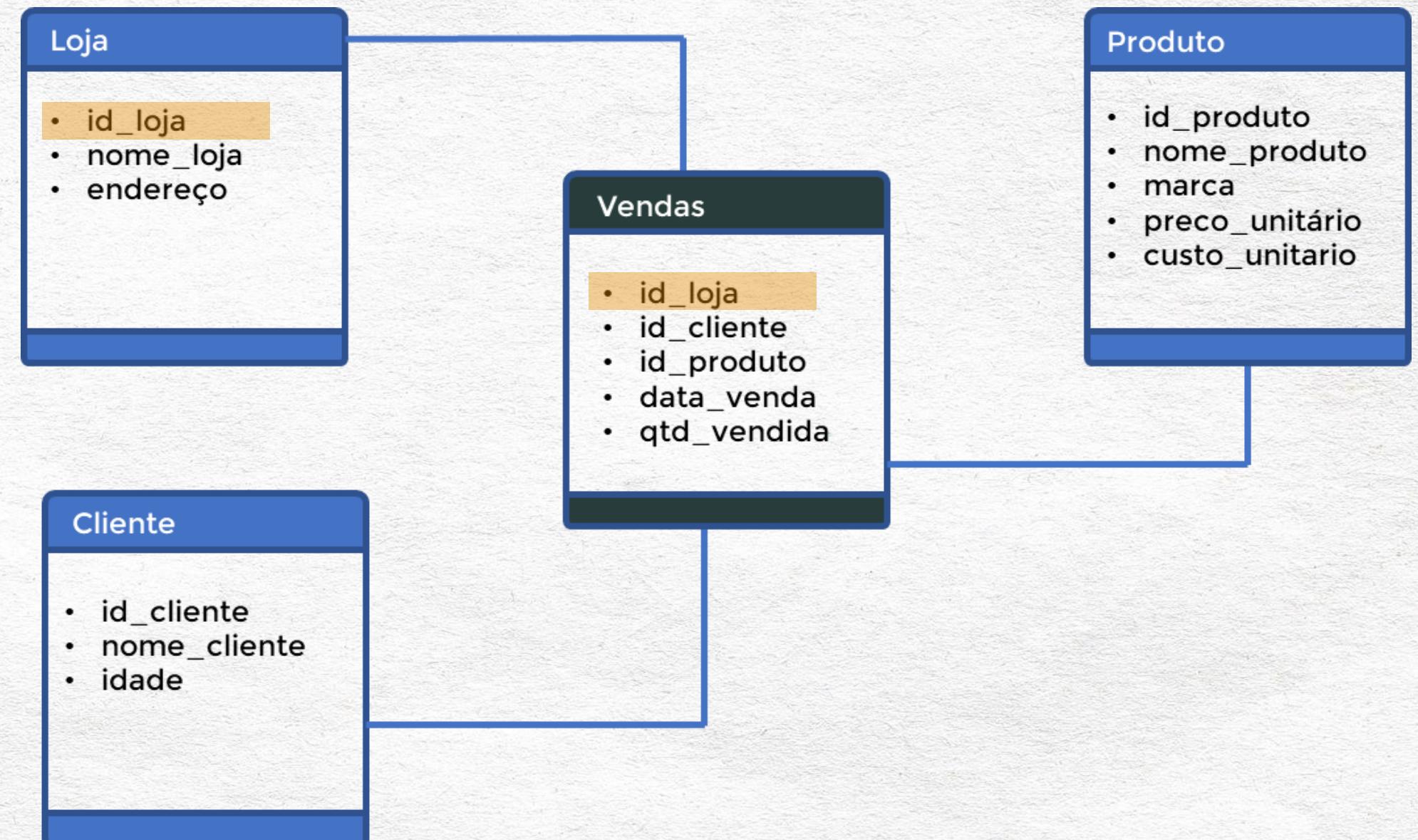


O que é um Banco de Dados?

O desenho esquemático de um banco de dados é mostrado ao lado. Diversas tabelas, com diferentes informações sobre um negócio, e que possuem algum tipo de relação entre si.

A esse banco de dados damos o nome de **RELACIONAL**.

Bancos de dados relacionais serão o foco do nosso curso, até por serem o tipo de bancos de dados mais comumente encontrados no mercado.

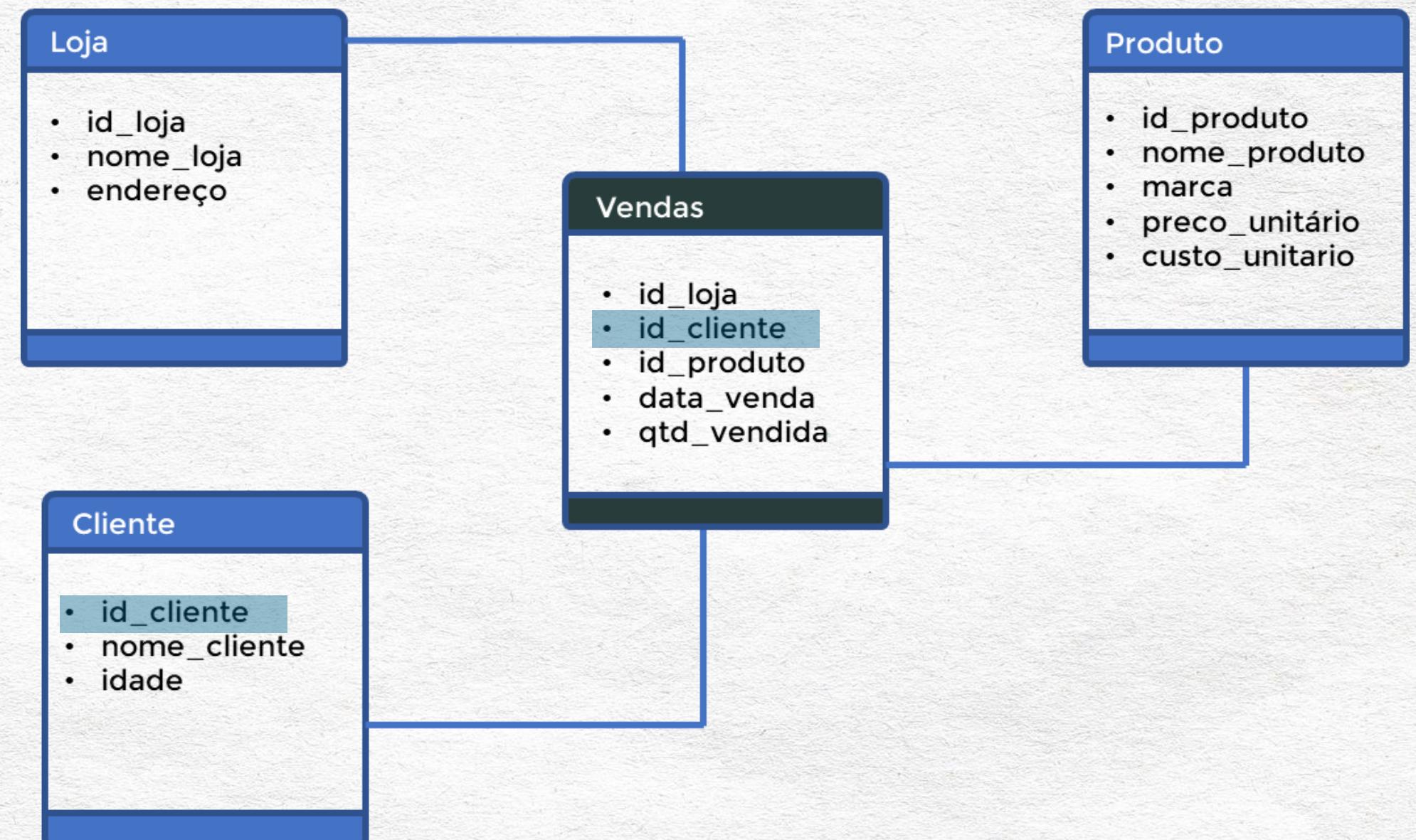


O que é um Banco de Dados?

O desenho esquemático de um banco de dados é mostrado ao lado. Diversas tabelas, com diferentes informações sobre um negócio, e que possuem algum tipo de relação entre si.

A esse banco de dados damos o nome de **RELACIONAL**.

Bancos de dados relacionais serão o foco do nosso curso, até por serem o tipo de bancos de dados mais comumente encontrados no mercado.

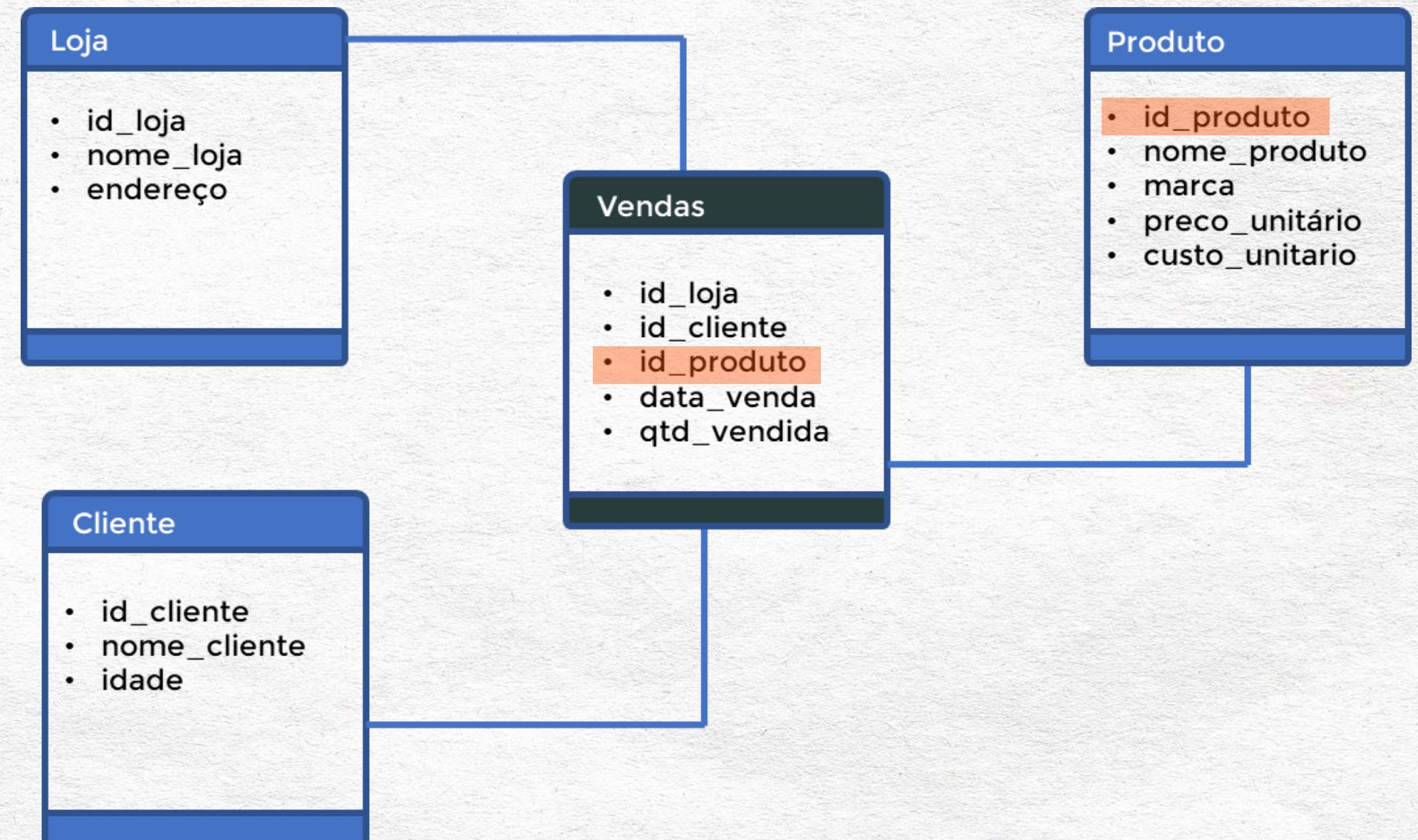


O que é um Banco de Dados?

O desenho esquemático de um banco de dados é mostrado ao lado. Diversas tabelas, com diferentes informações sobre um negócio, e que possuem algum tipo de relação entre si.

A esse banco de dados damos o nome de **RELACIONAL**.

Bancos de dados relacionais serão o foco do nosso curso, até por serem o tipo de bancos de dados mais comumente encontrados no mercado.



Sistemas de Bancos de Dados

014

Existem alguns SGBDs para Bancos de Dados Relacionais que são muito utilizados por grandes empresas. Abaixo, temos os 4 principais programas para SGBDs.

É importante que fique claro que **todos esses SGBDs** utilizam o SQL como linguagem de programação.



SQL: Structured Query Language

Para acessar e consultar os dados em um banco de dados, é necessário o uso de uma série de comandos.

Esses comandos, na verdade, se tratam de uma linguagem de programação, chamada SQL: Structured Query Language.

Traduzindo para o português, a sigla SQL significa Linguagem de Consulta Estruturada.

Essa é uma linguagem de bancos de dados universal e é por dela que será possível a consulta aos dados dentro dos bancos de dados.

SQL: Structured Query Language

Para acessar e consultar os dados em um banco de dados, é necessário o uso de uma série de comandos.

Esses comandos, na verdade, se tratam de uma linguagem de programação, chamada SQL: Structured Query Language.

Traduzindo para o português, a sigla SQL significa Linguagem de Consulta Estruturada.

Essa é uma linguagem de bancos de dados universal e é por dela que será possível a consulta aos dados dentro dos bancos de dados.

“

SQL significa “**Structured Query Language**”. Se trata de uma linguagem de programação utilizada para armazenar, consultar, adicionar e excluir informações em um banco de dados.

SQL: Structured Query Language

Para acessar e consultar os dados em um banco de dados, é necessário o uso de uma série de comandos.

Esses comandos, na verdade, se tratam de uma linguagem de programação, chamada SQL: Structured Query Language.

Traduzindo para o português, a sigla SQL significa Linguagem de Consulta Estruturada.

Essa é uma linguagem de bancos de dados universal e é por dela que será possível a consulta aos dados dentro dos bancos de dados.



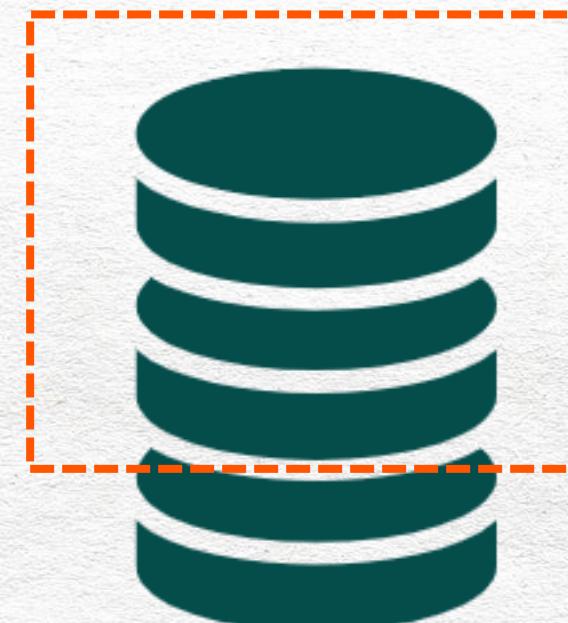
SQL: Structured Query Language

Para acessar e consultar os dados em um banco de dados, é necessário o uso de uma série de comandos.

Esses comandos, na verdade, se tratam de uma linguagem de programação, chamada SQL: Structured Query Language.

Traduzindo para o português, a sigla SQL significa Linguagem de Consulta Estruturada.

Essa é uma linguagem de bancos de dados universal e é por dela que será possível a consulta aos dados dentro dos bancos de dados.



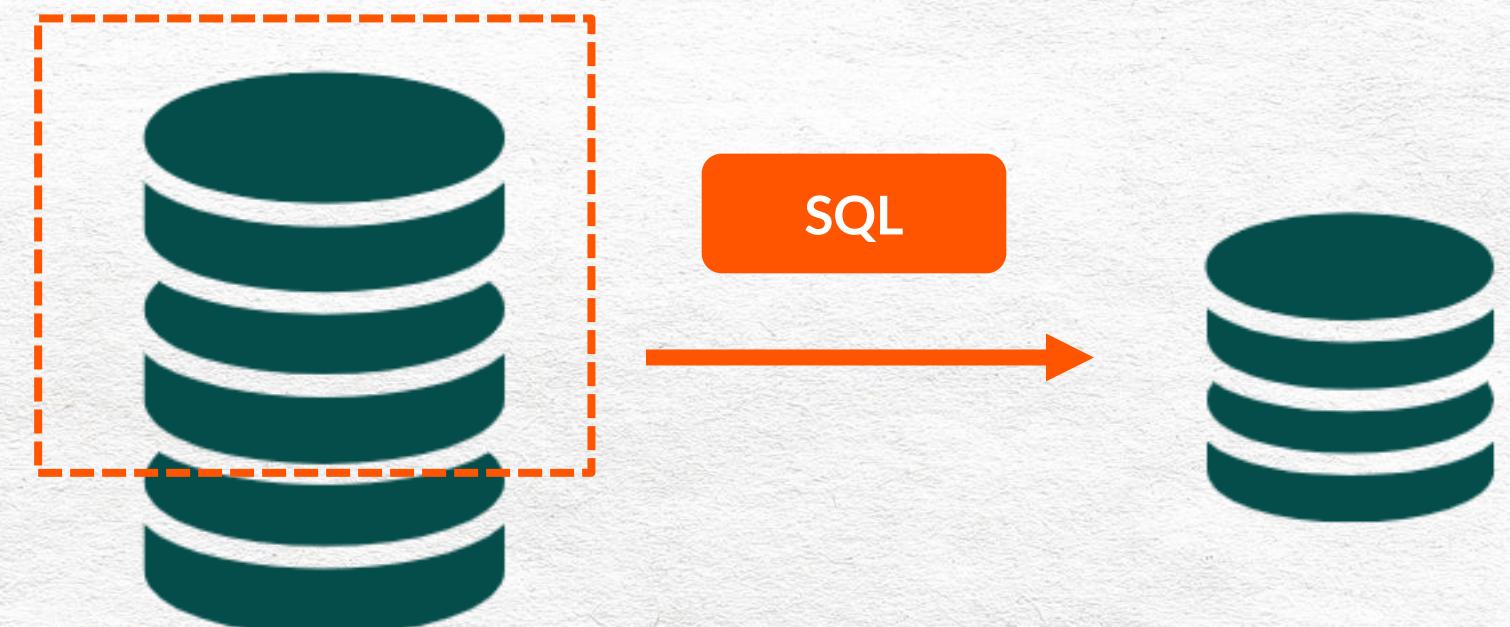
SQL: Structured Query Language

Para acessar e consultar os dados em um banco de dados, é necessário o uso de uma série de comandos.

Esses comandos, na verdade, se tratam de uma linguagem de programação, chamada SQL: Structured Query Language.

Traduzindo para o português, a sigla SQL significa Linguagem de Consulta Estruturada.

Essa é uma linguagem de bancos de dados universal e é por dela que será possível a consulta aos dados dentro dos bancos de dados.



O que é uma query (consulta)?

Uma query é um pedido de uma informação ou de um dado. Esse pedido também pode ser entendido como uma consulta, uma solicitação ou, ainda, uma requisição.

Em resumo, uma query (ou consulta) é uma leitura dos dados de uma tabela dentro de um banco de dados. Ou seja, quando queremos visualizar determinados dados de uma tabela, na prática o que queremos é fazer uma consulta aos dados do banco de dados.

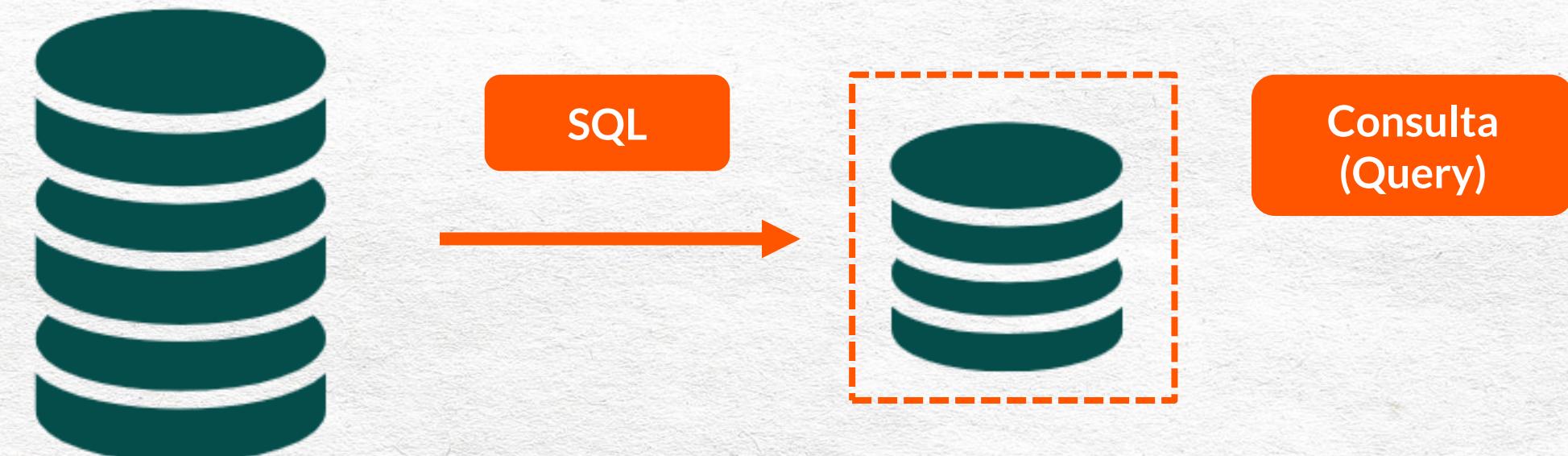
Porém, a leitura desses dados não é aleatória. Ela é baseada em uma série de comandos, feitos a partir da linguagem SQL.

O que é uma query (consulta)?

Uma query é um pedido de uma informação ou de um dado. Esse pedido também pode ser entendido como uma consulta, uma solicitação ou, ainda, uma requisição.

Em resumo, uma query (ou consulta) é uma leitura dos dados de uma tabela dentro de um banco de dados. Ou seja, quando queremos visualizar determinados dados de uma tabela, na prática o que queremos é fazer uma consulta aos dados do banco de dados.

Porém, a leitura desses dados não é aleatória. Ela é baseada em uma série de comandos, feitos a partir da linguagem SQL.



O que é uma query (consulta)?

Ao lado, temos um exemplo bem simplificado de como seria essa query (consulta) aos dados em uma tabela de um banco de dados, utilizando um código em SQL.

Tabela

id	produto	data_venda	valor
1	televisão	2021-03-20	1500
2	computador	2021-03-22	2300
3	celular	2021-03-25	800
4	ps4	2021-03-28	3100
5	tablet	2021-03-28	650

SQL

```
1 • SELECT *
2 FROM tabela_vendas
3 WHERE valor > 1000;
```



id	produto	data_venda	valor
1	televisão	2021-03-20	1500
2	computador	2021-03-22	2300
4	ps4	2021-03-28	3100

Consulta

Grupos de comando no SQL

Os comandos do SQL podem ser divididos em **quatro grupos**.

DDL

Data Definition Language

CREATE

Cria uma nova tabela, view ou outro objeto dentro do banco de dados.

ALTER

Modifica um objeto dentro do banco de dados (tabela, view, etc).

DROP

Exclui um objeto dentro do banco de dados (tabela, view, etc).

DML

Data Manipulation Language

INSERT

Adiciona uma nova linha em uma tabela.

UPDATE

Atualiza os valores das linhas de uma tabela.

DELETE

Exclui linhas de uma tabela.

DCL

Data Control Language

GRANT

Dá privilégios a um usuário.

REVOKE

Retira privilégios de um usuário.

DQL

Data Query Language

SELECT

Comando de seleção de linhas de uma tabela.

SQL, MySQL, SQL Server, Oracle e PostgreSQL: Não confunda os SQL's!!

O SQL é uma linguagem de consulta a bancos de dados, enquanto o MySQL, SQL Server, Oracle Database e PostgreSQL são programas utilizados para gerenciamento dos bancos de dados.

SQL, MySQL, SQL Server, Oracle e PostgreSQL: Não confunda os SQL's!!

O SQL é uma linguagem de consulta a bancos de dados, enquanto o MySQL, SQL Server, Oracle Database e PostgreSQL são programas utilizados para gerenciamento dos bancos de dados.

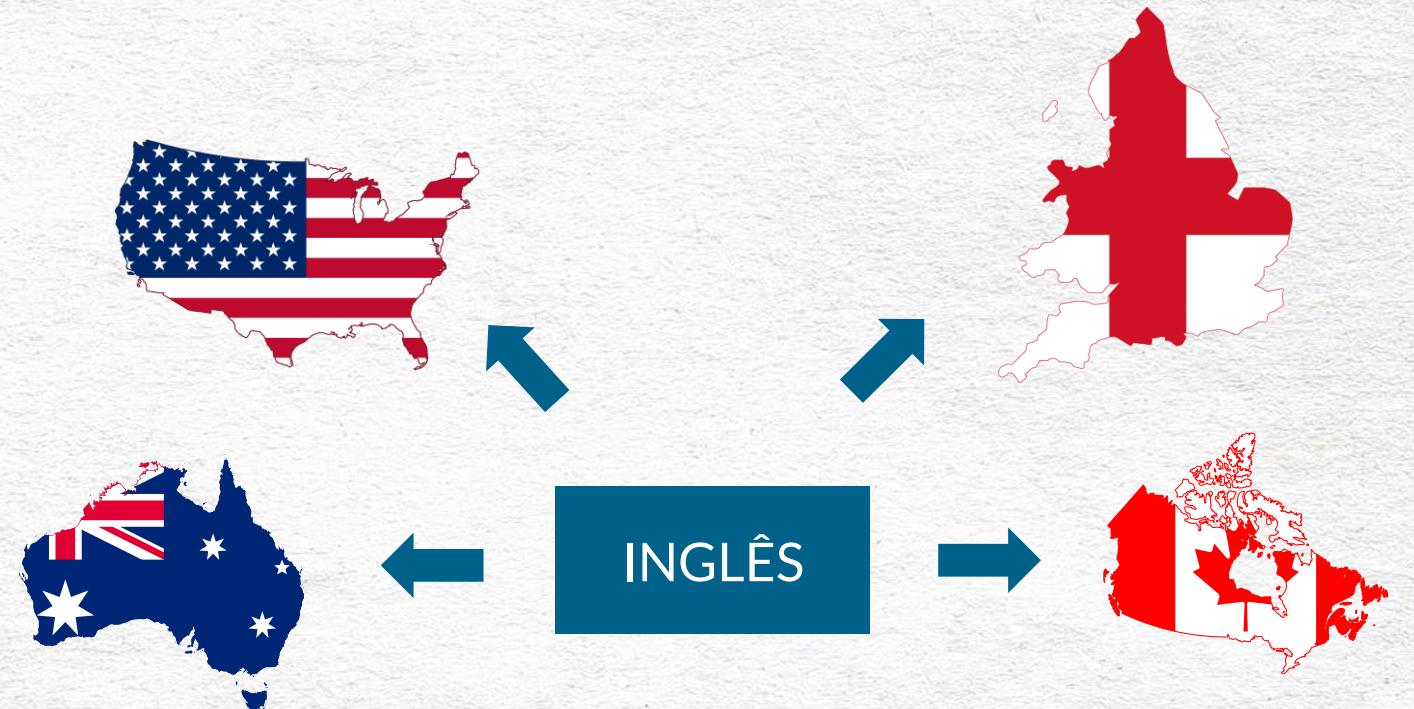
Para ficar clara essa diferença, faça um paralelo com o inglês.

SQL, MySQL, SQL Server, Oracle e PostgreSQL: Não confunda os SQL's!!

O SQL é uma linguagem de consulta a bancos de dados, enquanto o MySQL, SQL Server, Oracle Database e PostgreSQL são programas utilizados para gerenciamento dos bancos de dados.

Para ficar clara essa diferença, faça um paralelo com o inglês.

O inglês pode ser falado em diferentes países, como o EUA, Inglaterra, Austrália, dentre outros. Porém, não aprendemos inglês aplicado aos EUA, ou inglês aplicado à Inglaterra. Inglês é inglês, e pode ser falado em diferentes países.



SQL, MySQL, SQL Server, Oracle e PostgreSQL: Não confunda os SQL's!!

O SQL é uma linguagem de consulta a bancos de dados, enquanto o MySQL, SQL Server, Oracle Database e PostgreSQL são programas utilizados para gerenciamento dos bancos de dados.

Para ficar clara essa diferença, faça um paralelo com o inglês.

O inglês pode ser falado em diferentes países, como o EUA, Inglaterra, Austrália, dentre outros. Porém, não aprendemos inglês aplicado aos EUA, ou inglês aplicado à Inglaterra. Inglês é inglês, e pode ser falado em diferentes países.

O mesmo vale para o SQL. O SQL é uma linguagem de consulta (idioma) que pode ser usado em diferentes programas (países): MySQL, SQL Server, Oracle e PostgreSQL.



SQL, MySQL, SQL Server, Oracle e PostgreSQL: Não confunda os SQL's!!

O SQL é uma linguagem de consulta a bancos de dados, enquanto o MySQL, SQL Server, Oracle Database e PostgreSQL são programas utilizados para gerenciamento dos bancos de dados.

Para ficar clara essa diferença, faça um paralelo com o inglês.

O inglês pode ser falado em diferentes países, como o EUA, Inglaterra, Austrália, dentre outros. Porém, não aprendemos inglês aplicado aos EUA, ou inglês aplicado à Inglaterra. Inglês é inglês, e pode ser falado em diferentes países.

O mesmo vale para o SQL. O SQL é uma linguagem de consulta (idioma) que pode ser usado em diferentes programas (países): MySQL, SQL Server, Oracle e PostgreSQL.

Portanto, não se assuste com tantos SQLs. Na verdade, só existe um SQL. Todos os demais são programas que utilizam o SQL para realizar as consultas aos bancos de dados.



MÓDULO 2

INSTALAÇÃO

INSTALAÇÃO

INSTALAÇÃO



Instalação do MySQL

O SGBD escolhido será o **MySQL**.

Lembrando que o SGBD será composto essencialmente por 2 partes: um **Servidor** e uma **Interface**.

Para o caso do MySQL, teremos que instalar o **MySQL Server** e o **MySQL Workbench**, respectivamente.

MySQL Server



um **servidor**, onde vamos conseguir armazenar os nossos bancos de dados.

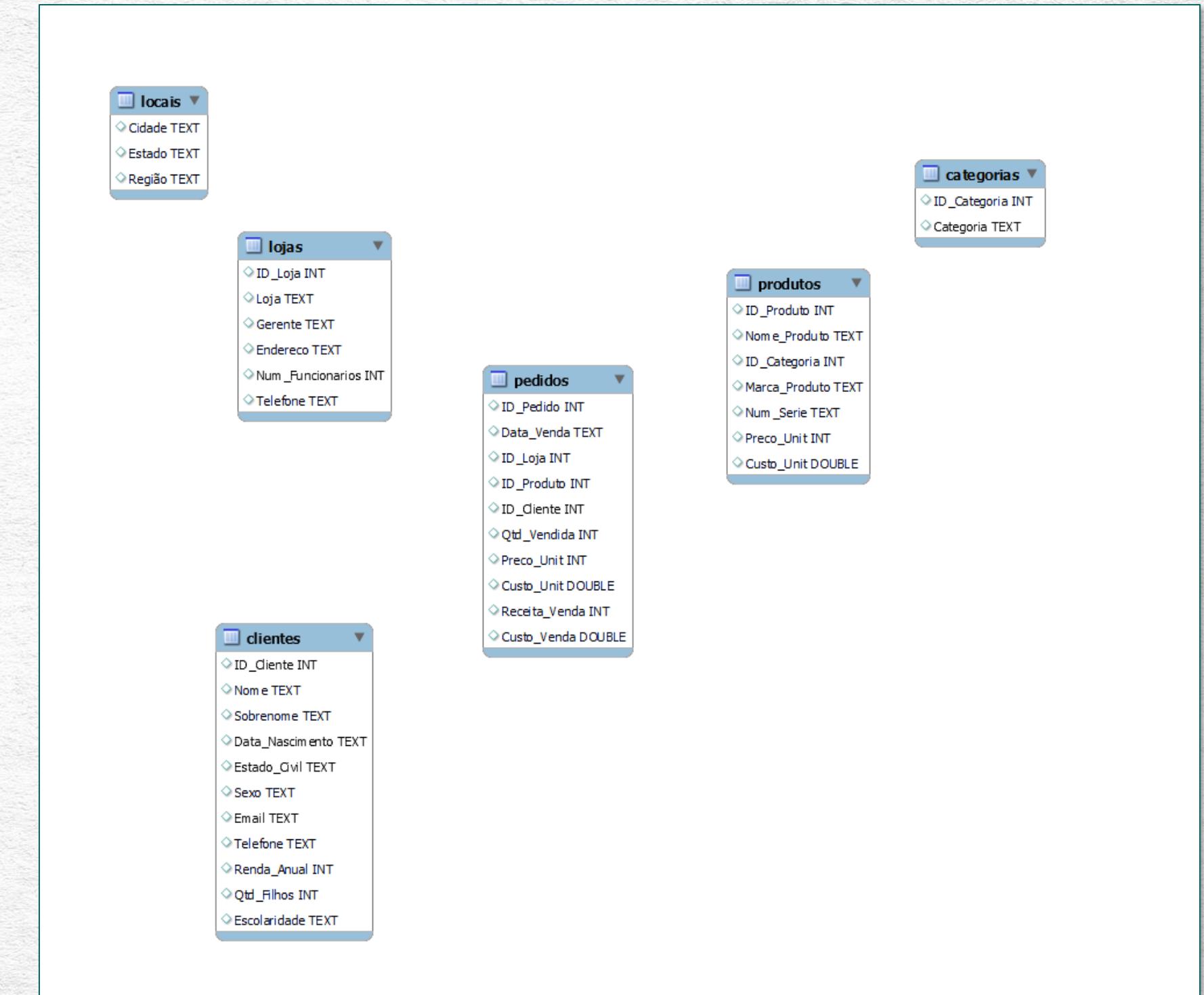
MySQL Workbench



uma **interface** amigável que nos permite escrever os códigos em SQL para acessar os bancos de dados.

Banco de Dados utilizado

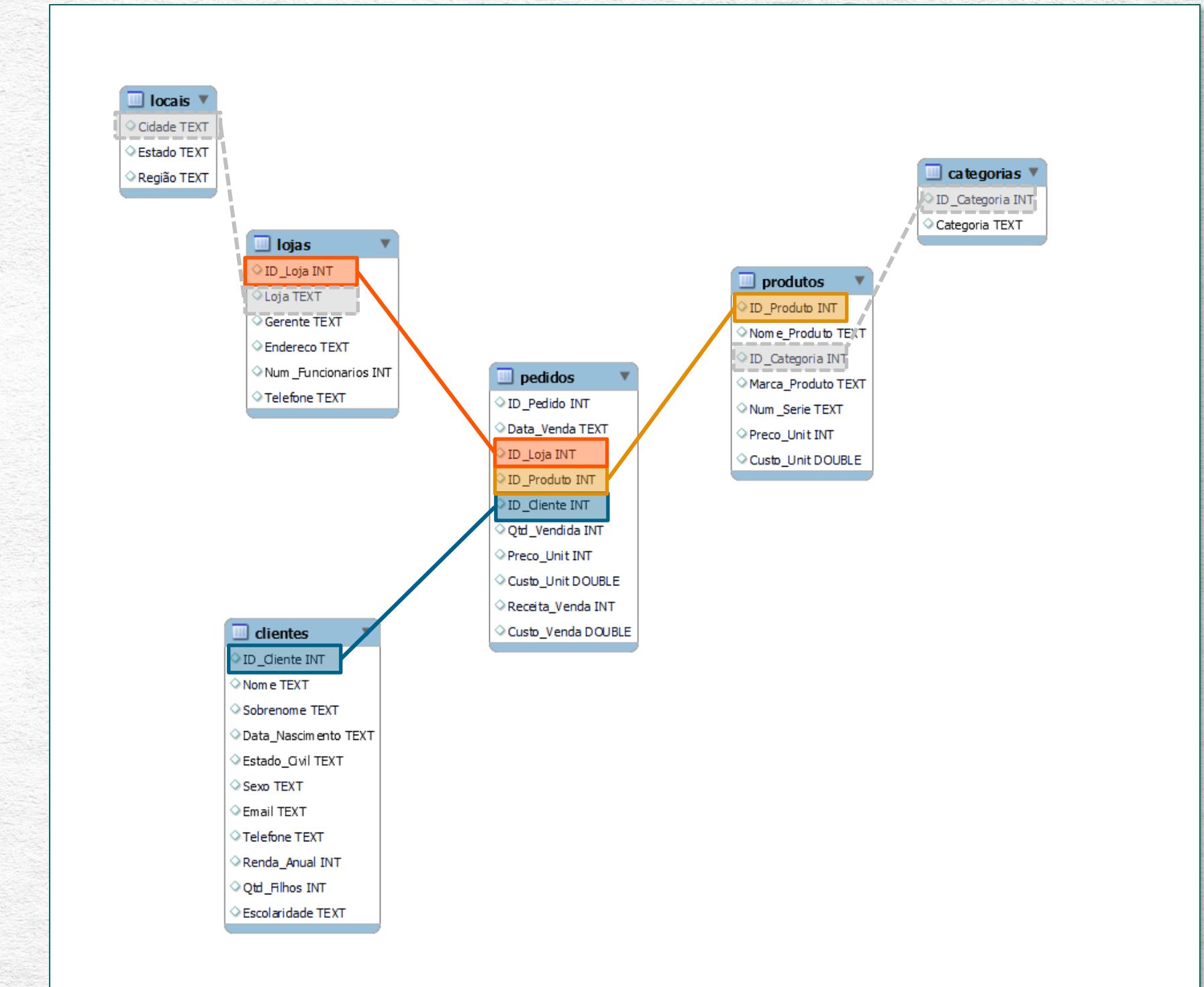
As tabelas do banco de dados estão mostradas na imagem ao lado.



Banco de Dados utilizado

As tabelas do banco de dados estão mostradas na imagem ao lado.

Existem colunas em comum entre as tabelas, para que seja possível criar um banco de dados relacional.



MÓDULO 3

CONSULTAS BÁSICAS

CONSULTAS BÁSICAS

CONSULTAS BÁSICAS

SELECT *, SELECT FROM, SELECT AS e SELECT LIMIT

034

Vamos começar aprendendo sobre os comandos de seleção de dados no MySQL. São eles:

Comandos de Seleção

SELECT FROM

SELECT LIMIT

SELECT AS

SELECT DISTINCT



SQL IMPRESSIONADOR | HASHTAG TREINAMENTOS

SELECT *

OBJETIVO

Selecionar **todas as colunas** e todas as linhas de uma tabela.

CÓDIGO

```
1 • SELECT *
2 FROM Tabela;
```

RESULTADO

Col1	Col2	Col3
1	Nome	10.5
2	Texto	5.45
3	Carro	51.3
4	Celular	100.6
5	SQL	500

SELECT (colunas)

OBJETIVO

Selecionar **apenas colunas específicas** de uma tabela.

CÓDIGO

```
1 • SELECT Col1, Col2  
2   FROM Tabela;
```

RESULTADO

Col1	Col2
1	Nome
2	Texto
3	Carro
4	Celular
5	SQL

SELECT AS

OBJETIVO

Selecionar colunas específicas e dar um nome para essas colunas.

CÓDIGO

```
1 • SELECT  
2     Col1 AS "Coluna 1",  
3     Col2 AS "Coluna 2"  
4 FROM Tabela;
```

RESULTADO

Coluna 1	Coluna 2
1	Nome
2	Texto
3	Carro
4	Celular
5	SQL

SELECT LIMIT

OBJETIVO

Selecionar apenas as N primeiras linhas de uma determinada tabela.

CÓDIGO

```
1 • SELECT  
2      *  
3 FROM Tabela  
4 LIMIT 2;
```

RESULTADO

Col1	Col2	Col3
1	Nome	10.5
2	Texto	5.45

SELECT DISTINCT

OBJETIVO

Selecionar apenas os valores distintos de uma coluna.

CÓDIGO

```
SELECT  
    DISTINCT genero  
FROM filmes;
```

RESULTADO

genero
Comédia
Drama
Ficção e Fantasia
Mistério e Suspense
Arte
Animação
Ação e Aventura

ORDER BY DESC e ORDER BY ASC

Os comandos de ordenação permitem a ordenação dos dados da nossa tabela, a partir de uma coluna. Com ele, podemos ordenar por ordem crescente, ordem alfabética, e assim vai.

Comandos de Ordenação

ORDER BY ASC

ORDER BY DESC

ORDER BY (DESC)

OBJETIVO

Permite ordenar (classificar) uma tabela a partir de uma determinada coluna em **ordem decrescente**.

CÓDIGO

```
1 • SELECT  
2      *  
3 FROM Tabela  
4 ORDER BY Col3 DESC;
```

RESULTADO

Col1	Col2	Col3
5	SQL	500
4	Celular	100.6
3	Carro	51.3
1	Nome	10.5
2	Texto	5.45

ORDER BY (ASC)

OBJETIVO

Permite ordenar (classificar) uma tabela a partir de uma determinada coluna em **ordem crescente**.

CÓDIGO

```
1 • SELECT  
2      *  
3 FROM Tabela  
4 ORDER BY Col3;
```

RESULTADO

Col1	Col2	Col3
2	Texto	5.45
1	Nome	10.5
3	Carro	51.3
4	Celular	100.6
5	SQL	500

Existem 3 formas de comentar códigos, assim como mostrado ao lado.

Obs: As opções de traço duplo e barra-asterisco funcionam em qualquer SGBD que usarmos.

```
1 -- Este é um comentário
2 /* um código qualquer
3
4 # Este é outro comentário
5 outro código qualquer
6
7 /* Este é um bloco de código
8   este é outro bloco de código */
9 outro código
10 mais um código
```

MÓDULO 4

FILTROS

FILTROS

FILTROS



WHERE: Comando para filtrar dados

Os comandos de filtragem nos permitem criar filtros nas nossas tabelas dos bancos de dados.

Comandos de Filtragem

WHERE

WHERE AND/OR

WHERE IN

WHERE BETWEEN

Essencialmente, podemos fazer filtros com colunas que contenham 3 tipos de informação.

Veremos também que é possível criar filtros em mais de 1 coluna ao mesmo tempo.

NÚMEROS

TEXTOS

DATAS



NÚMEROS

Podemos aplicar filtros em colunas numéricas.

Para isso, basta utilizar os sinais lógicos como =, <, >, <=, >=, <>.

```
1
2 -- Mostra apenas os produtos com preços iguais ou maiores que R$1.800
3 • SELECT *
4 FROM produtos
5 WHERE Preco_Unit >= 1800;
```

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160



TEXTOS

Podemos aplicar filtros em colunas de texto.

Para isso, basta utilizar o sinal de = e especificar o texto que deseja usar como critério do filtro (ou o <> caso queira apenas o que for diferente de).

```
1  
2 -- Mostra apenas os produtos da marca DELL  
3 • SELECT *  
4 FROM produtos  
5 WHERE Marca_Produto = 'DELL';
```

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176



DATAS

Podemos aplicar filtros em colunas de data.

Para isso, basta utilizar os sinais lógicos como =, <, >, <=, >=, <>.

```
1  
2 -- Mostra apenas os pedidos feitos no dia 03/01/2019  
3 • SELECT *  
4 FROM pedidos  
5 WHERE Data_Venda = '2019-01-03';
```

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Receita_Venda	Custo_Venda	Custo_Unit	Preco_Unit
16	2019-01-03	8	4	33	1	350	129.5	129.5	350
17	2019-01-03	8	4	36	1	350	129.5	129.5	350
18	2019-01-03	1	4	37	1	350	129.5	129.5	350
19	2019-01-03	6	4	42	1	350	129.5	129.5	350
20	2019-01-03	8	4	43	1	350	129.5	129.5	350
21	2019-01-03	6	4	44	1	350	129.5	129.5	350
88	2019-01-03	6	8	29	1	600	258	258	600
89	2019-01-03	5	8	46	1	600	258	258	600
90	2019-01-03	2	8	86	1	600	258	258	600
91	2019-01-03	4	8	12	1	600	258	258	600
92	2019-01-03	4	8	82	1	600	258	258	600
93	2019-01-03	3	8	41	1	600	258	258	600



OPERADOR AND

Podemos aplicar mais de um filtro usando o **AND**.

Com ele, todas as condições devem ser satisfeitas para que o resultado seja mostrado.

```
1  
2 -- Mostra apenas os clientes SOLTEIROS do sexo MASCULINO  
3 • SELECT *  
4 FROM clientes  
5 WHERE Estado_Civil = 'S' AND Sexo = 'M';
```

ID_Cliente	Nome	Sobrenome	Data_Nascimento	Estado_Civil	Sexo	Email	Telefone	Renda_Anual	Qtd_Filhos	Escolaridade
4	Julio	Ruiz	1985-07-31	S	M	julio1@hotmail.com	(62) 93391-5891	70000	0	Pós-graduado
8	Shannon	Carlson	1984-03-27	S	M	shannon38@gmail.com	(85) 96795-9950	70000	0	Pós-graduado
17	Clarence	Rai	1964-10-04	S	M	clarence32@outlook.com	(21) 98923-7805	30000	2	Parcial
18	Jordan	King	1998-09-15	S	M	jordan73@gmail.com	(31) 99592-6279	40000	0	Ensino médio
24	Harold	Sai	1966-03-29	S	M	harold3@hotmail.com	(31) 95052-6286	30000	2	Parcial
43	Leonard	Nara	1970-05-14	S	M	leonard18@outlook.com	(31) 93245-4616	30000	3	Ensino médio
49	Daniel	Johnson	1971-07-30	S	M	daniel18@gmail.com	(31) 98422-3549	30000	3	Ensino médio
65	Caleb	Carter	1996-09-20	S	M	caleb40@gmail.com	(31) 97809-1800	60000	0	Parcial
72	Levi	Arun	1976-08-23	S	M	levi6@gmail.com	(71) 92754-9983	70000	2	Ensino médio
74	Blake	Anderson	1977-07-08	S	M	blake9@gmail.com	(11) 98232-2736	80000	2	Ensino médio
77	Donald	Gonzalez	1979-03-06	S	M	donald20@gmail.com	(31) 96897-9735	160000	0	Graduação
82	Lucas	Phillips	1977-09-07	S	M	lucas7@outlook.com	(62) 94668-3507	80000	2	Parcial
88	Trevor	Bryant	1977-12-12	S	M	trevor18@gmail.com	NULL	90000	2	Parcial
92	Cedric	Ma	1982-03-27	S	M	cedric15@outlook.com	(31) 95423-4764	70000	1	Parcial
93	Chad	Kumar	1982-08-27	S	M	chad9@gmail.com	(21) 98828-7409	70000	1	Parcial



OPERADOR OR

Podemos aplicar mais de um filtro usando o **OR**.

Com ele, apenas uma condição precisa ser satisfeita para que o resultado seja mostrado.

```
1
2 -- Mostra apenas os produtos das marcas DELL OU SAMSUNG
3 • SELECT *
4 FROM produtos
5 WHERE Marca_Produto = 'DELL' OR Marca_Produto = 'SAMSUNG';
```

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160



OPERADOR IN

Com o operador IN, podemos passar uma lista de valores que serão utilizados como filtro na nossa coluna.

```
86 -- Seleccione apenas os clientes das seguintes nacionalidades:  
87 -- Canadá, Porto Rico e Irlanda.  
88  
89 • SELECT *  
90 FROM atores  
91 WHERE nacionalidade IN ('Canadá', 'Porto Rico', 'Irlanda');  
92
```

Result Grid Filter Rows: [] Export: [] Wrap Cell Content: []					
	id_ator	nome_ator	ano_nascimento	nacionalidade	sexo
	11	Barry Pepper	1970	Canadá	masculino
	12	Benicio Del Toro	1967	Porto Rico	masculino
	29	Colin Farrell	1976	Irlanda	masculino
	35	Donald Sutherland	1935	Canadá	masculino
	81	Keanu Reeves	1964	Canadá	masculino
	112	Rachel McAdams	1978	Canadá	feminino
	134	Stephen Rea	1946	Irlanda	masculino



BETWEEN

Com o operador IN, podemos passar uma lista de valores que serão utilizados como filtro na nossa coluna.

```
86 -- Seleciona apenas os clientes que nasceram entre os anos de:  
87 -- 1980 e 1983.
```

```
88  
89 • SELECT *  
90 FROM atores  
91 WHERE ano_nascimento BETWEEN 1980 AND 1983;  
92
```

	id_ator	nome_ator	ano_nascimento	nacionalidade	sexo
▶	1	Abbie Cornish	1982	Austrália	feminino
	5	Andrea Riseborough	1981	Reino Unido	feminino
	8	Anne Hathaway	1982	EUA	feminino
	18	Bryce Dallas Howard	1981	EUA	feminino
	22	Channing Tatum	1980	EUA	masculino
	26	Christina Ricci	1980	EUA	feminino
	37	Eddie Redmayne	1982	Reino Unido	masculino
	40	Elijah Wood	1981	EUA	masculino
	70	Jesse Eisenberg	1983	EUA	masculino
	84	Kristen Bell	1980	EUA	feminino
	94	Maryam Karimi	1980	Afeganistão	feminino
	102	Natalie Portman	1981	Israel	feminino
	124	Sam Riley	1980	Reino Unido	masculino
	145	Zooey Deschanel	1980	EUA	feminino



LIKE e NOT LIKE

Com o operador LIKE, podemos filtrar textos que contenham determinada cadeia de caracteres.

```
1 -- WHERE LIKE: filtro de 'contém'  
2 • SELECT *  
3 FROM clientes  
4 WHERE Email LIKE '%gmail%';
```

ID_Cliente	Nome	Sobrenome	Data_Nascimento	Estado_Civil	Sexo	Email	Telefone	Renda_Anual	Qtd_Filhos	Escolaridade
3	Elizabeth	Johnson	1988-08-03	S	F	elizabeth5@gmail.com	(31) 92039-5832	80000	5	Pós-graduado
5	Janet	Alvarez	1985-12-01	S	F	janet9@gmail.com	(21) 93379-3743	70000	0	Pós-graduado
8	Shannon	Carlson	1984-03-27	S	M	shannon38@gmail.com	(85) 96795-9950	70000	0	Pós-graduado
12	Ian	Jenkins	1988-08-01	C	M	ian47@gmail.com	(21) 95385-3393	100000	2	Pós-graduado
13	Sydney	Bennett	1988-05-04	S	F	sydney23@gmail.com	(11) 92024-9186	100000	3	Pós-graduado
15	Wyatt	Hill	1999-04-23	C	M	wyatt32@gmail.com	(71) 98644-2695	70000	0	Parcial
18	Jordan	King	1998-09-15	S	M	jordan73@gmail.com	(31) 99591-0000	100000	0	Ensino médio
21	Seth	Edwards	1998-10-06	C	M	seth46@gmail.com	(41) 97012-3526	40000	0	Parcial
23	Alejandro	Beck	1965-12-18	C	M	alejandro45@gmail.com	(85) 94492-9582	10000	2	Parcial
26	Jill	Jimenez	1966-04-06	C	F	jill13@gmail.com	(41) 97120-5603	30000	2	Parcial
32	Ebony	Gonzalez	1967-06-14	C	F	ebony19@gmail.com	(71) 99664-2156	20000	4	Ensino médio
33	Wendy	Dominguez	1968-02-19	C	F	wendy12@gmail.com	(31) 97278-7135	10000	2	Parcial
35	Chloe	Garcia	1997-11-22	S	F	chloe27@gmail.com	(21) 94134-4552	40000	0	Parcial
36	Diana	Hernandez	1968-03-18	C	F	diana2@gmail.com	(85) 93094-3166	10000	2	Parcial

MÓDULO 5

O P E R A Ç Õ E S B Á S I C A S E
F U N Ç Õ E S D E A G R E G A Ç Ã O

O P E R A Ç Õ E S B Á S I C A S E
F U N Ç Õ E S D E A G R E G A Ç Ã O

O P E R A Ç Õ E S B Á S I C A S E
F U N Ç Õ E S D E A G R E G A Ç Ã O

Operações matemáticas

É muito simples fazer operações matemáticas no SQL. Ao lado, temos uma sequência de exemplos que vão desde a soma até o cálculo do resto de uma divisão, usando o operador %.

```

1 -- Operações matemáticas
2 • SELECT
3      10 + 20      AS 'Soma',
4      150 - 50      AS 'Subtração',
5      300 * 3      AS 'Multiplicação',
6      450 / 5      AS 'Divisão',
7      (2 + 3) * 5  AS 'Combinação',
8      22 % 5       AS 'Resto da divisão'

```

	Soma	Subtração	Multiplicação	Divisão	Combinação	Resto da divisão
▶	30	100	900	90.0000	25	2

As funções matemáticas são extremamente importantes para criação de **cálculos no SQL**. Eles serão a base para as análises que faremos com agrupamentos e GROUP BY!

Funções Matemáticos

COUNT

SUM

MIN/MAX

AVG

Funções matemáticas

Essas funções têm como objetivo realizar cálculos no SQL, tais como: contagem, soma, média, mínimo e máximo.

COUNT

COUNT(*)

COUNT(DISTINCT)

SUM

AVG

MIN/MAX

COUNT

OBJETIVO

Retorna a **quantidade total de valores** de uma coluna.

CÓDIGO

```
1   -- COUNT  
2 • SELECT  
3       COUNT(Nome)  
4 FROM clientes;
```

RESULTADO

	COUNT(Nome)
▶	100

COUNT

OBJETIVO

Retorna a **quantidade total de valores** de uma coluna.

CÓDIGO

```
1 -- COUNT  
2 • SELECT  
3     COUNT(Telefone)  
4 FROM clientes;
```

RESULTADO

	COUNT(Telefone)
▶	94



O COUNT ignora os valores nulos de uma coluna. Por isso o resultado pode mudar dependendo da coluna escolhida.

COUNT(*)

OBJETIVO

Retorna a **quantidade total de linhas** de uma tabela.

Obs: não ignora valores nulos

CÓDIGO

```
1   -- COUNT  
2 • SELECT  
3       COUNT(*)  
4   FROM clientes;
```

RESULTADO

	COUNT(*)
▶	100

COUNT(DISTINCT)

OBJETIVO

Retorna a **contagem distinta** de valores de uma tabela.

CÓDIGO

```
1 -- COUNT(DISTINCT)
2 • SELECT
3     COUNT(DISTINCT Escolaridade)
4 FROM clientes;
```

RESULTADO

COUNT(DISTINCT Escolaridade)
4

SUM

OBJETIVO

Retorna a **soma total** dos valores de uma coluna.

CÓDIGO

```
1  -- SUM  
2 • SELECT  
3      SUM(Receita_Venda)  
4  FROM pedidos;
```

RESULTADO

SUM(Receita_Venda)
228900

AVG

OBJETIVO

Retorna a **média dos valores** de uma coluna.

CÓDIGO

```
1 -- AVG  
2 • SELECT  
3     AVG(Receita_Venda)  
4 FROM pedidos;
```

RESULTADO

	AVG(Receita_Venda)
▶	612.0321

MAX e MIN

OBJETIVO

Retorna o **valor máximo e mínimo** de uma coluna.

CÓDIGO

```
1 -- MAX
2 • SELECT
3     MAX(Receita_Venda)
4 FROM pedidos;
```

RESULTADO

	MAX(Receita_Venda)
▶	1800

MÓDULO 6

AGRUPAMENTOS

AGRUPAMENTOS

AGRUPAMENTOS

GROUP BY

O comando GROUP BY permite agrupar os dados de uma tabela, criando uma espécie de resumo dos dados.

Data Venda	Produto	Quantidade
2019-01-01	PRODUTO A	10
2019-01-01	PRODUTO B	20
2019-01-01	PRODUTO A	5
2019-01-01	PRODUTO A	15
2019-01-01	PRODUTO B	10
2019-01-01	PRODUTO A	30

GROUP BY



Produto	Quantidade Total
PRODUTO A	60
PRODUTO B	30

GROUP BY

O comando GROUP BY permite agrupar os dados de uma tabela, criando uma espécie de resumo dos dados.

Data Venda	Produto	Quantidade
2019-01-01	PRODUTO A	10
2019-01-01	PRODUTO B	20
2019-01-01	PRODUTO A	5
2019-01-01	PRODUTO A	15
2019-01-01	PRODUTO B	10
2019-01-01	PRODUTO A	30

GROUP BY



Produto	Quantidade Total
PRODUTO A	60
PRODUTO B	30

GROUP BY

O comando GROUP BY permite agrupar os dados de uma tabela, criando uma espécie de resumo dos dados.

Data Venda	Produto	Quantidade
2019-01-01	PRODUTO A	10
2019-01-01	PRODUTO B	20
2019-01-01	PRODUTO A	5
2019-01-01	PRODUTO A	15
2019-01-01	PRODUTO B	10
2019-01-01	PRODUTO A	30

GROUP BY



Produto	Quantidade Total
PRODUTO A	60
PRODUTO B	30

GROUP BY

O comando GROUP BY permite agrupar os dados de uma tabela, criando uma espécie de resumo dos dados.

Data Venda	Produto	Quantidade
2019-01-01	PRODUTO A	10
2019-01-01	PRODUTO B	20
2019-01-01	PRODUTO A	5
2019-01-01	PRODUTO A	15
2019-01-01	PRODUTO B	10
2019-01-01	PRODUTO A	30

GROUP BY



```
SELECT Produto,  
       SUM(Quantidade) AS  
             'Quantidade Total'  
  FROM Tabela  
 GROUP BY Produto;
```

Produto	Quantidade Total
PRODUTO A	60
PRODUTO B	30



GROUP BY

O **Group By** é o comando do SQL que vai nos permitir **criar agrupamentos**, ou seja, tabelas resumos das nossas tabelas principais.

```
1 # GROUP BY  
2  
3 -- Exemplo: Utilize o GROUP BY para criar uma  
4 -- consulta e descobrir o total de filmes por gênero.  
5 • SELECT
```

```
6     genero,  
7     COUNT(*)  
8 FROM filmes  
9 GROUP BY genero;
```

genero	COUNT(*)
Comédia	10
Drama	36
Ficção e Fantasia	11
Mistério e Suspense	6
Arte	1
Animação	2
Ação e Aventura	5

Wrap Cell Content:

FILTRANDO AGRUPAMENTOS (WHERE)



GROUP BY + WHERE

Sempre que quisermos realizar um filtro **ANTES** de criar o agrupamento, usamos o WHERE.

```
1 # GROUP BY + WHERE
2
3 -- Exemplo: Utilize o GROUP BY + WHERE para criar uma
4 -- consulta e descobrir o total de filmes por gênero, mas
5 -- considerando apenas os filmes lançados em 2003.
6 • SELECT
7     genero,
8     COUNT(*)
9 FROM filmes
10 WHERE ano_lancamento = 2003
11 GROUP BY genero;
```

genero	COUNT(*)
Mistério e Suspense	2
Comédia	2
Drama	3
Ficção e Fantasia	1



FILTRANDO AGRUPAMENTOS (HAVING)



GROUP BY + HAVING

Sempre que quisermos realizar um filtro **DEPOIS** que o agrupamento foi criado, usamos o HAVING.

```

86 # GROUP BY + HAVING
87
88 -- Exemplo: Utilize o GROUP BY + HAVING para filtrar a tabela resultante e mostrar
89 -- apenas os filmes dos gêneros com quantidade MAIOR OU IGUAL A 10 filmes.
90
91 • SELECT
92     genero,
93     COUNT(*) AS 'qtd_filmes'
94 FROM filmes
95 GROUP BY genero
96 HAVING COUNT(*) >= 10;
97
98

```

	genero	qtd_filmes
▶	Comédia	10
	Drama	36
	Ficção e Fantasia	11

WHERE vs HAVING

Como o **WHERE** funciona?

1

Partimos da tabela **original**

id_filme	titulo	genero	duracao	ano_lancamento	preco_aluguel
1	Que mulher é essa?	Comédia	93	2001	2.09
2	A Senha	Drama	99	2001	2.19
3	Do que as mulheres gostam	Comédia	127	2001	2.59
4	Dia de Treinamento	Drama	122	2001	1.79
5	O Senhor dos Anéis: A sociedade do anel	Ficção e Fantasia	178	2001	2.59
6	Harry Potter e a Pedra Filosofal	Ficção e Fantasia	152	2001	2.69
7	Os Excêntricos Tenenbaums	Comédia	110	2002	1.89
8	Seu marido e minha mulher	Comédia	91	2002	2.59
9	11 de setembro	Drama	134	2002	2.99
10	Simone	Drama	117	2002	2.69
11	É hora do show	Comédia	95	2002	1.79
12	O Senhor dos Anéis: As duas torres	Ficção e Fantasia	179	2002	2.39
13	Harry Potter e a Câmara Secreta	Ficção e Fantasia	161	2002	1.79
14	O Novato	Mistério e Suspense	115	2003	1.69
15	Alguém tem que ceder	Comédia	128	2003	1.69
16	A última noite	Drama	135	2003	1.59
17	Revelações	Mistério e Suspense	106	2003	1.99
18	Lições Para Toda a Vida	Drama	111	2003	1.69
19	21 gramas	Drama	124	2003	2.09
20	Simplesmente amor	Comédia	135	2003	2.29
21	O Senhor dos Anéis: O retorno do rei	Ficção e Fantasia	200	2003	1.99
22	Visões	Arte	107	2004	2.59
23	Dança comigo	Drama	106	2004	1.69
24	Uma Eleição Muito Atrapalhada	Comédia	110	2004	2.89
25	Bridget Jones: No Limite da Razão	Drama	108	2004	2.89
26	Ray	Drama	152	2004	2.59
27	Monster - Desejo Assassino	Drama	109	2004	2.09
28	Harry Potter e o Prisioneiro de Azkaban	Ficção e Fantasia	142	2004	1.69
29	Tudo por Dinheiro	Drama	122	2005	2.79
30	Capote	Drama	114	2005	2.39
31	Harry Potter e o Cálice de Fogo	Ficção e Fantasia	157	2005	2.69
32	Falsária	Drama	93	2006	2.89
33	V de Vingança	Drama	132	2006	1.59
34	Armações do Amor	Drama	97	2006	1.99
35	Happy Feet	Animação	108	2006	1.79
36	As Torres Gêmeas	Drama	129	2006	1.59

```
SELECT
    genero,
    COUNT(*)
FROM filmes
WHERE ano_lancamento = 2003
GROUP BY genero;
```

2

Usamos o GROUP BY + WHERE

id_filme	titulo	genero	duracao	ano_lancamento	preco_aluguel
1	Que mulher é essa?	Comédia	93	2001	2.09
2	A Senha	Drama	99	2001	2.19
3	Do que as mulheres gostam	Comédia	127	2001	2.59
4	Dia de Treinamento	Drama	122	2001	1.79
5	O Senhor dos Anéis: A sociedade do anel	Ficção e Fantasia	178	2001	2.59
6	Harry Potter e a Pedra Filosofal	Ficção e Fantasia	152	2001	2.69
7	Os Excêntricos Tenenbaums	Comédia	110	2002	1.89
8	Seu marido e minha mulher	Comédia	91	2002	2.59
9	11 de setembro	Drama	134	2002	2.99
10	Simone	Drama	117	2002	2.69
11	É hora do show	Comédia	95	2002	1.79
12	O Senhor dos Anéis: As duas torres	Ficção e Fantasia	179	2002	2.39
13	Harry Potter e a Câmara Secreta	Ficção e Fantasia	161	2002	1.79
14	O Novato	Mistério e Suspense	115	2003	1.69
15	Alguém tem que ceder	Comédia	128	2003	1.69
16	A última noite	Drama	135	2003	1.59
17	Revelações	Mistério e Suspense	106	2003	1.99
18	Lições Para Toda a Vida	Drama	111	2003	1.69
19	21 gramas	Drama	124	2003	2.09
20	Simplesmente amor	Comédia	135	2003	2.29
21	O Senhor dos Anéis: O retorno do rei	Ficção e Fantasia	200	2003	1.99
22	Visões	Arte	107	2004	2.59
23	Dança comigo	Drama	106	2004	1.69
24	Uma Eleição Muito Atrapalhada	Comédia	110	2004	2.89
25	Bridget Jones: No Limite da Razão	Drama	108	2004	2.89
26	Ray	Drama	152	2004	2.59
27	Monster - Desejo Assassino	Drama	109	2004	2.09
28	Harry Potter e o Prisioneiro de Azkaban	Ficção e Fantasia	142	2004	1.69
29	Tudo por Dinheiro	Drama	122	2005	2.79
30	Capote	Drama	114	2005	2.39
31	Harry Potter e o Cálice de Fogo	Ficção e Fantasia	157	2005	2.69
32	Falsária	Drama	93	2006	2.89
33	V de Vingança	Drama	132	2006	1.59
34	Armações do Amor	Drama	97	2006	1.99
35	Happy Feet	Animação	108	2006	1.79
36	As Torres Gêmeas	Drama	129	2006	1.59

3

Apenas os filmes do ano de 2003 são considerados

E o agrupamento final de gêneros considera apenas os filmes de 2003

genero	COUNT(*)
Mistério e Suspense	2
Comédia	2
Drama	3
Ficção e Fantasia	1

WHERE vs HAVING

Como o **HAVING** funciona?

1

Partimos da tabela **agrupada**

	genero	COUNT(*)
▶	Comédia	10
	Drama	36
	Ficção e Fantasia	11
	Mistério e Suspense	6
	Arte	1
	Animação	2
	Ação e Aventura	5

2

Usamos o GROUP BY + HAVING

```
SELECT
    genero,
    COUNT(*) AS 'qtd_filmes'
FROM filmes
GROUP BY genero
HAVING COUNT(*) >= 10;
```

3

Apenas os gêneros com
quantidade MAIOR OU IGUAL
são considerados

	genero	COUNT(*)
▶	Comédia	10
	Drama	36
	Ficção e Fantasia	11
	Mistério e Suspense	6
	Arte	1
	Animação	2
	Ação e Aventura	5

4

E o agrupamento final de
gêneros considera apenas os
gêneros com mais de 10 filmes

	genero	qtd_filmes
▶	Comédia	10
	Drama	36
	Ficção e Fantasia	11

MÓDULO 7

VARIÁVEIS

VARIÁVEIS

VARIÁVEIS



O que é uma variável

Uma variável é um local onde armazenamos um determinado valor, que pode ser usado ao longo do nosso código.

Por exemplo, observe a query abaixo:

```
• SELECT *
  FROM produtos
 WHERE Marca_Produto = 'SAMSUNG';
```

O que é uma variável

Uma variável é um local onde armazenamos um determinado valor, que pode ser usado ao longo do nosso código.

Poderíamos utilizar variáveis para facilitar na organização e compreensão do código.

```
SET @varMarca = 'SAMSUNG';

SELECT *
FROM produtos
WHERE Marca_Produto = @varMarca;
```

Escopo das Variáveis

O escopo de uma variável se refere ao local onde essa variável existe, ou seja, o local onde ela pode ser acessada.

Temos os seguintes níveis de escopo:

1. USER-DEFINED VARIABLES
2. LOCAL VARIABLES

Tipos de Dados no MySQL

080

Na tabela abaixo listamos os principais tipos de dados usados para variáveis.

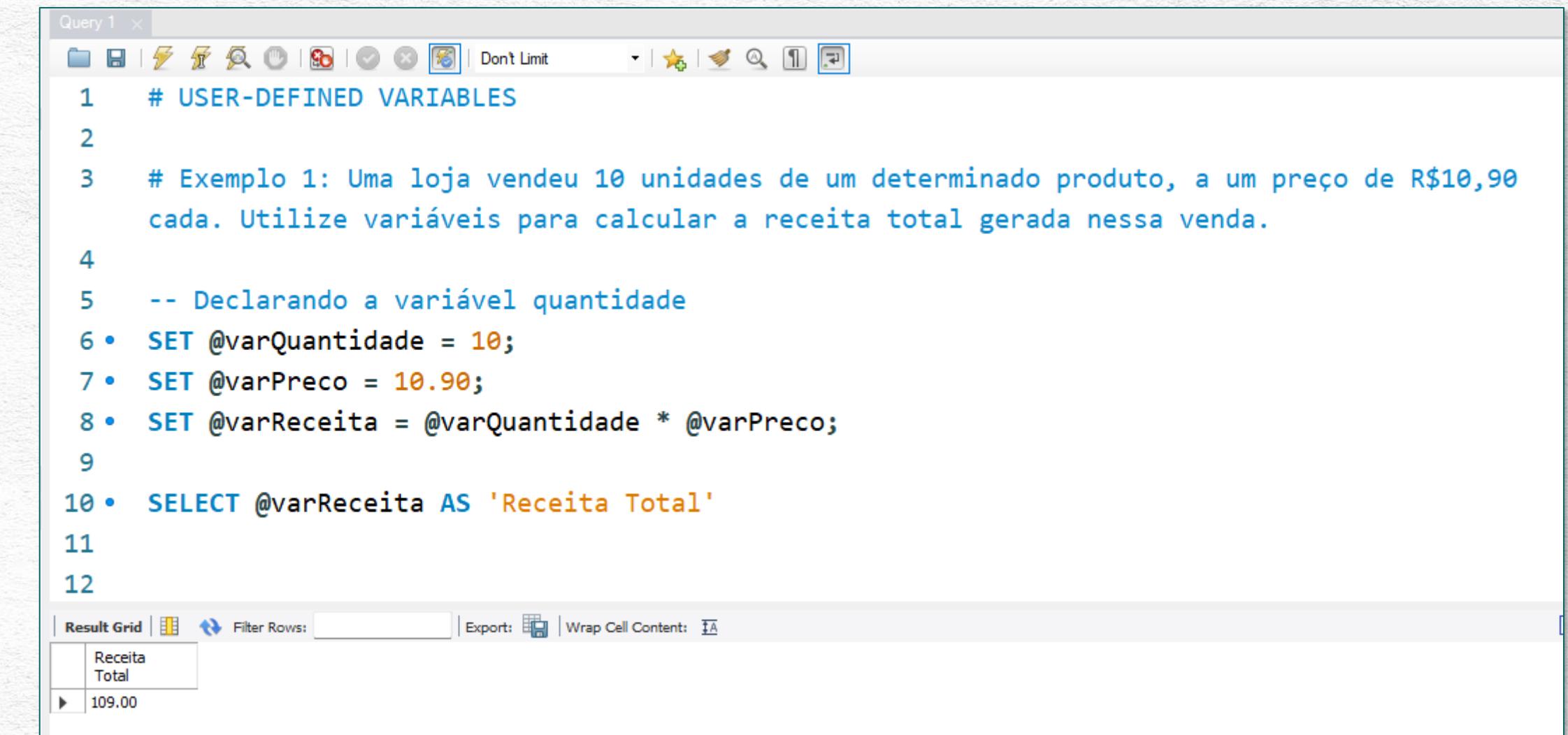
TIPO	DESCRIÇÃO
INT	Inteiros entre -2.147.483.648 e 2.147.483.647
DECIMAL(M, D)	Número decimal com M dígitos no total e D casas decimais. O padrão é (10, 2). Exemplo: o valor 100,34 poderia ser declarado como DECIMAL(5, 2).
FLOAT(M, D)	Número decimal com M dígitos no total e D casas decimais. O padrão é (10, 2). Exemplo: o valor 12345,67 poderia ser declarado como FLOAT(7, 2).
VARCHAR(N)	String de tamanho variável, até 65535 caracteres.
DATE	Uma data de 01/01/1000 a 31/12/9999, no formato YYYY-MM-DD
DATETIME	Uma combinação de data e hora de 01/01/1000 00:00:00 a 31/12/9999 23:59:59, no formato YYYY-MM-DD HH:MM:SS

User-Defined Variables na Prática (Exemplo 1)

Vamos agora ver na prática como vão funcionar as variáveis.

Ao lado, temos a seguinte situação: cálculo da receita total de uma venda, a partir da quantidade vendida e do preço do produto.

A solução final pode ser observada na imagem ao lado.



The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The code in the editor is:

```
1 # USER-DEFINED VARIABLES
2
3 # Exemplo 1: Uma loja vendeu 10 unidades de um determinado produto, a um preço de R$10,90
4 # cada. Utilize variáveis para calcular a receita total gerada nessa venda.
5 -- Declarando a variável quantidade
6 • SET @varQuantidade = 10;
7 • SET @varPreco = 10.90;
8 • SET @varReceita = @varQuantidade * @varPreco;
9
10 • SELECT @varReceita AS 'Receita Total'
11
12
```

The results grid below the editor shows a single row with the following data:

Receita Total
109.00

User-Defined Variables na Prática (Exemplo 2)

32

Podemos aplicar variáveis para facilitar consultas ao banco de dados.

Exemplo: imagine que você deseja filtrar a tabela de produtos de acordo com a Marca, mas usando uma variável para isso.

Ao lado, temos a solução para esta situação.

13 # Exemplo 2: Faça uma consulta na tabela de produtos de forma que você visualize apenas os produtos da marca DELL. Utilize variáveis para que a marca possa ser facilmente alterada dentro da variável.

```
14  
15 • SET @varMarca = 'DELL';
```

```
16  
17 •   SELECT *  
18     FROM produtos  
19     WHERE Marca_Pro
```

26

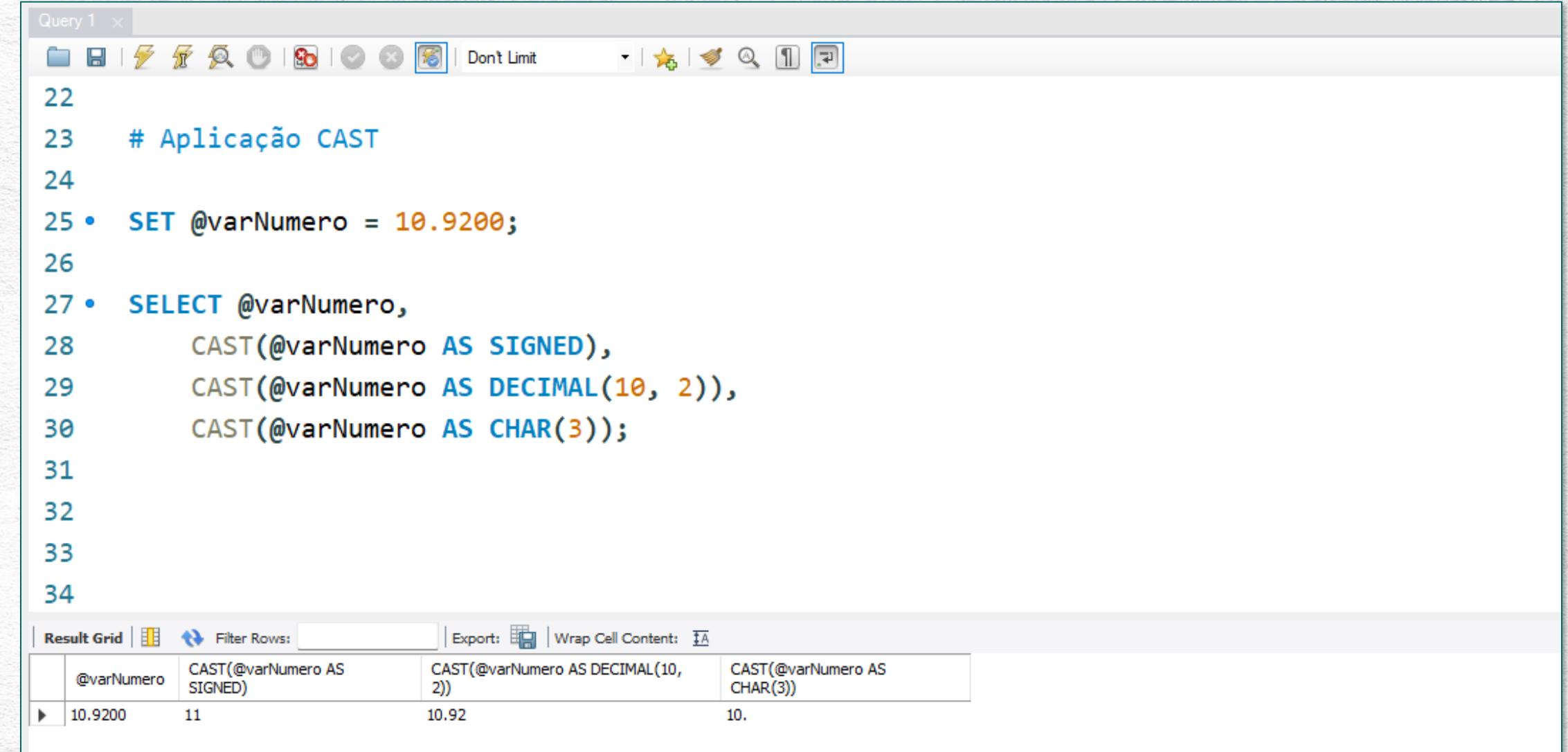
Result Grid		Filter Rows:		Export:	Wrap Cell Content:			
	ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit	
▶	1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966	
	4	Kit Tedado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5	
	5	Kit Tedado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2	
	14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209	
	15	Notebook IdeaPad 32000	6	DELL	NOT-DL-77164T	4200	1176	

Função CAST

A função CAST converte um valor de qualquer tipo em um outro tipo de dados especificado.

No CAST, especificamos o INT como SIGNED (valores positivos e negativos) ou UNSIGNED (somente valores positivos) e o VARCHAR como CHAR.

Ao lado, temos um exemplo de aplicação da função CAST sobre um número armazenado na variável @varNumero.



The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The code in the editor is as follows:

```
22
23  # Aplicação CAST
24
25 • SET @varNumero = 10.9200;
26
27 • SELECT @varNumero,
28     CAST(@varNumero AS SIGNED),
29     CAST(@varNumero AS DECIMAL(10, 2)),
30     CAST(@varNumero AS CHAR(3));
31
32
33
34
```

The result grid below the code shows the output of the query:

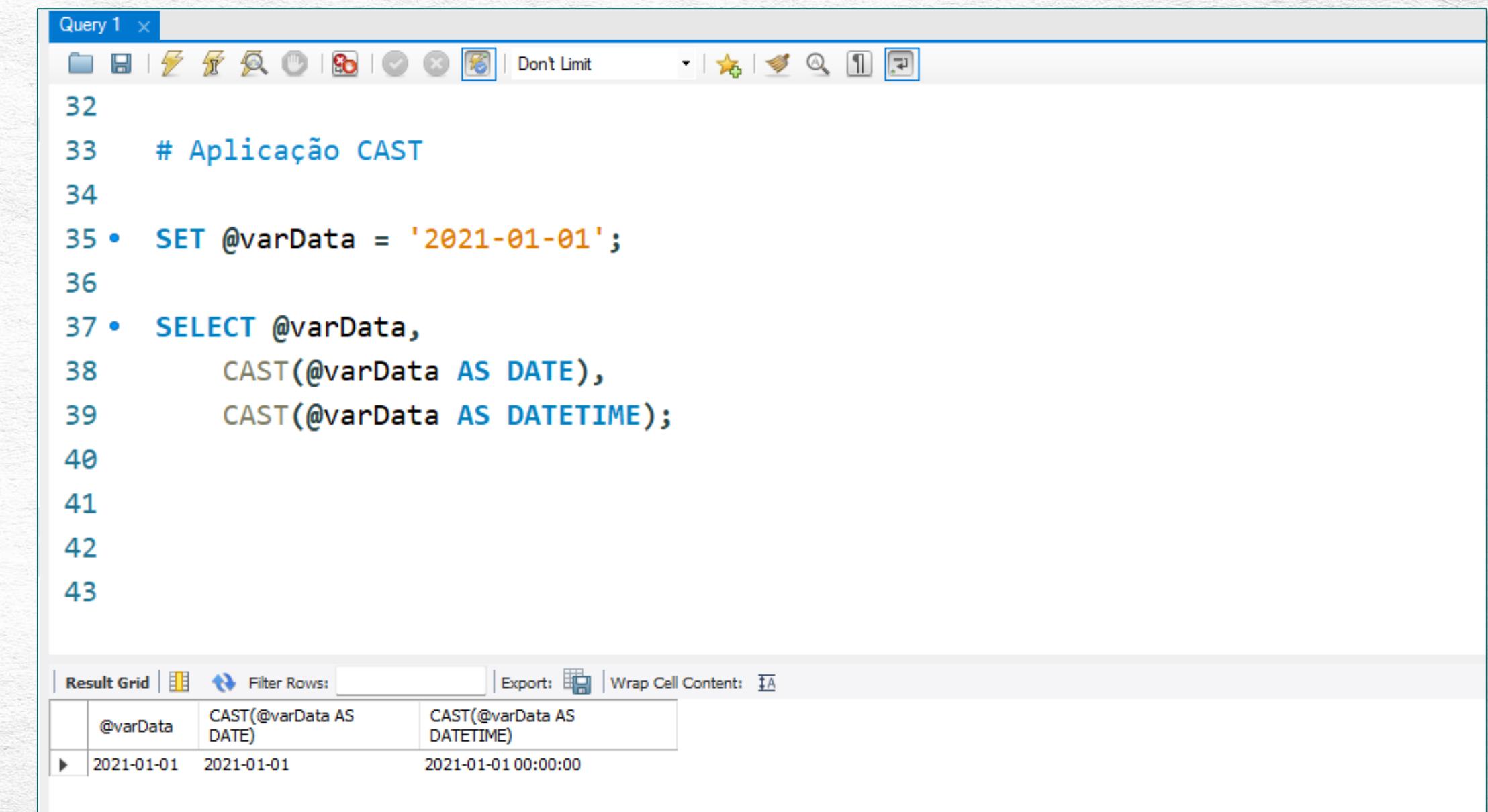
@varNumero	CAST(@varNumero AS SIGNED)	CAST(@varNumero AS DECIMAL(10, 2))	CAST(@varNumero AS CHAR(3))
10.9200	11	10.92	10.

Função CAST

A função CAST converte um valor de qualquer tipo em um outro tipo de dados especificado.

No CAST, especificamos o INT como SIGNED (valores positivos e negativos) ou UNSIGNED (somente valores positivos) e o VARCHAR como CHAR.

Ao lado, temos um exemplo de aplicação da função CAST sobre um número armazenado na variável @varNumero.



The screenshot shows a MySQL Workbench interface with a query editor titled 'Query 1'. The code is as follows:

```
32
33  # Aplicação CAST
34
35 • SET @varData = '2021-01-01';
36
37 • SELECT @varData,
38      CAST(@varData AS DATE),
39      CAST(@varData AS DATETIME);
40
41
42
43
```

The result grid shows the output of the query:

	@varData	CAST(@varData AS DATE)	CAST(@varData AS DATETIME)
▶	2021-01-01	2021-01-01	2021-01-01 00:00:00

MÓDULO 8

FUNÇÕES DE TEXTO E DATA

FUNÇÕES DE TEXTO E DATA

FUNÇÕES DE TEXTO E DATA

Funções de Texto e Data

As funções de texto (string) e data nos permitem fazer diversos tipos de tratamentos com valores do tipo texto e do tipo data.

Abaixo, temos um resumo das principais:

STRINGS

Funções para trabalhar com **TEXTOS**.

- LENGTH
- CONCAT
- CONCAT_WS
- LCASE e UCASE
- RIGHT e LEFT
- REPLACE
- INSTR
- MID

DATAS

Funções para trabalhar com **DATAS**.

- DAY
- YEAR
- MONTH
- NOW
- CURDATE
- CURTIME
- DATE_DIFF
- DATE_ADD
- DATE_SUB



Vamos começar com as funções de texto (string).

STRINGS

Funções para trabalhar
com **TEXTOS**.

- LENGTH
- CONCAT
- CONCAT_WS
- LCASE e UCASE
- RIGHT e LEFT
- REPLACE
- INSTR
- MID

DATAS

Funções para trabalhar
com **DATAS**.

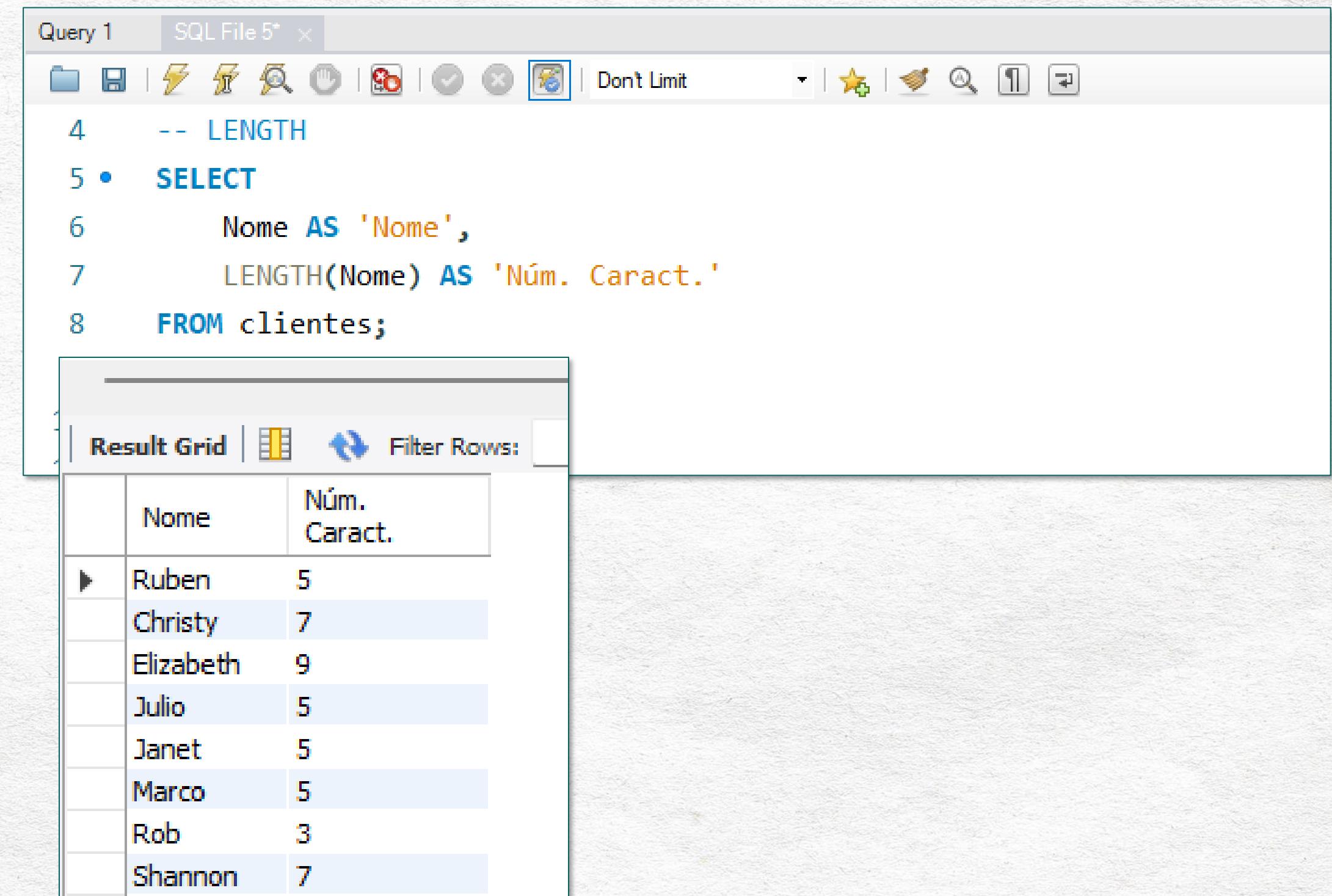
- DAY
- YEAR
- MONTH
- NOW
- CURDATE
- CURTIME
- DATE_DIFF
- DATE_ADD
- DATE_SUB



Função LENGTH

A função LENGTH pede apenas 1 argumento e retorna a quantidade de caracteres que um determinado texto possui.

Por exemplo, o nome Ruben possui 5 caracteres, portanto este é o retorno da função.



The screenshot shows a MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
Query 1 SQL File 5* ×
4 -- LENGTH
5 • SELECT
6     Nome AS 'Nome',
7     LENGTH(Nome) AS 'Núm. Caract.'
8 FROM clientes;
```

The result grid displays the following data:

	Nome	Núm. Caract.
▶	Ruben	5
	Christy	7
	Elizabeth	9
	Julio	5
	Janet	5
	Marco	5
	Rob	3
	Shannon	7

Função CONCAT e CONCAT_WS

As funções CONCAT e CONCAT_WS retornam a concatenação de textos.

A diferença entre elas é que na CONCAT_WS passamos no primeiro argumento o caractere que desejamos usar para separar cada texto.

No CONCAT, cada vez que precisamos de um espaço para separar os textos, precisaremos repetir o espaço diversas vezes.

```

12 -- CONCAT e CONCAT_WS
13 • SELECT
14     CONCAT(Nome, ' ', Sobrenome) AS 'Nome Completo (Concat)',
15     CONCAT_WS(' ', Nome, Sobrenome) AS 'Nome Completo (Concat_ws)'
16 FROM clientes;
17

```

	Nome Completo (Concat)	Nome Completo (Concat_ws)
▶	Ruben Torres	Ruben Torres
	Christy Zhu	Christy Zhu
	Elizabeth Johnson	Elizabeth Johnson
	Julio Ruiz	Julio Ruiz
	Janet Alvarez	Janet Alvarez
	Marco Mehta	Marco Mehta
	Rob Verhoff	Rob Verhoff
	Shannon Carlson	Shannon Carlson
	Jacquelyn Suarez	Jacquelyn Suarez
	Curtis Lu	Curtis Lu

Função LCASE e UCASE

Estas funções permitem retornar textos em maiúscula (UCASE) e minúscula (LCASE).

Ao lado, temos um exemplo de aplicação dessas duas funções para retornar o nome completo de cada cliente tanto em maiúscula quanto em minúscula.

```

18    -- LCASE e UCASE
19 •  SELECT
20      UCASE(CONCAT(Nome, ' ', SobreNome)) AS 'Nome Completo (Concat)',
21      LCASE(CONCAT_WS(' ', Nome, SobreNome)) AS 'Nome Completo (Concat_ws)'
22  FROM clientes;
23

```

	Nome Completo (Concat)	Nome Completo (Concat_ws)
RUBEN TORRES	ruben torres	ruben torres
CHRISTY ZHU	christy zhu	christy zhu
ELIZABETH JOHNSON	elizabeth johnson	elizabeth johnson
JULIO RUIZ	julio ruiz	julio ruiz
JANET ALVAREZ	janet alvarez	janet alvarez
MARCO MEHTA	marco mehta	marco mehta
ROB VERHOFF	rob verhoff	rob verhoff
SHANNON CARLSON	shannon carlson	shannon carlson
JACQUELYN SUAREZ	jacquelyn suarez	jacquelyn suarez
CURTIS LU	curtis lu	curtis lu
LAUREN WALKER	lauren walker	lauren walker
IAN JENKINS	ian jenkins	ian jenkins
SYDNEY BENNETT	sydney bennett	sydney bennett
CHLOE YOUNG	chloe young	chloe young
WYATT HILL	wyatt hill	wyatt hill
SHANNON WANG	shannon wang	shannon wang
CLARENCE RAI	clarence rai	clarence rai
JORDAN KING	iordan kind	iordan kind

Função LEFT e RIGHT

As funções LEFT e RIGHT permitem retornar parte de um texto.

A função LEFT pede 2 argumentos: o primeiro é o texto, o segundo é a quantidade de caracteres desejada, **da esquerda para a direita**.

Já a função RIGHT faz o oposto: permite retornar uma determinada quantidade de caracteres, **da direita para a esquerda**.

```

26    -- LEFT e RIGHT
27 • SELECT
28     Num_Serie AS 'Serie',
29     LEFT(Num_Serie, 6) AS 'Cod 1',
30     RIGHT(Num_Serie, 6) AS 'Cod 2'
31 FROM produtos;
32

```

	Serie	Cod 1	Cod 2
▶	MNT-DL-831923	MNT-DL	831923
	MNT-SS-001939	MNT-SS	001939
	WBC-LT-934GG4	WBC-LT	934GG4
	KTM-DL-041039	KTM-DL	041039
	KTM-DL-111924	KTM-DL	111924
	CGM-AL-9N914J	CGM-AL	9N914J
	CGM-AL-0147FI	CGM-AL	0147FI
	HDP-SN-194821	HDP-SN	194821
	HDP-JB-091934	HDP-JB	091934
	MIC-AK-237591	MIC-AK	237591
	MIC-BL-819455	MIC-BL	819455
	MIC-BL-761411	MIC-BL	761411
	NOT-SS-918457	NOT-SS	918457
	NOT-DL-000012	NOT-DL	000012
	NOT-DL-77164I	NOT-DL	77164I
	NOT-SS-13139U	NOT-SS	13139U

Função LEFT e RIGHT

Ao lado, aplicamos essas duas funções para dividir a número de série de cada um dos produtos em duas partes.

Por exemplo, o número de Série:

MNT-DL-831923

Pode ser dividido em 2 partes usando as funções LEFT e RIGHT.

MNT-DL 831923

LEFT

RIGHT

```

26    -- LEFT e RIGHT
27 •  SELECT
28      Num_Serie AS 'Serie',
29      LEFT(Num_Serie, 6) AS 'Cod 1',
30      RIGHT(Num_Serie, 6) AS 'Cod 2'
31  FROM produtos;
32

```

	Serie	Cod 1	Cod 2
▶	MNT-DL-831923	MNT-DL	831923
	MNT-SS-001939	MNT-SS	001939
	WBC-LT-934GG4	WBC-LT	934GG4
	KTM-DL-041039	KTM-DL	041039
	KTM-DL-111924	KTM-DL	111924
	CGM-AL-9N914J	CGM-AL	9N914J
	CGM-AL-0147FI	CGM-AL	0147FI
	HDP-SN-194821	HDP-SN	194821
	HDP-JB-091934	HDP-JB	091934
	MIC-AK-237591	MIC-AK	237591
	MIC-BL-819455	MIC-BL	819455
	MIC-BL-761411	MIC-BL	761411
	NOT-SS-918457	NOT-SS	918457
	NOT-DL-000012	NOT-DL	000012
	NOT-DL-77164I	NOT-DL	77164I
	NOT-SS-13139U	NOT-SS	13139U

Função REPLACE

A função REPLACE é outra função bem comum em diversas ferramentas. Ela permite substituir um determinado texto por outro texto.

Ao lado, temos a seguinte aplicação: utilizamos a função REPLACE para substituir as siglas do estado civil pelo texto correspondente.

Por exemplo, substituir S por Solteiro.

```

34  -- REPLACE
35 • SELECT * FROM clientes;
36 • SELECT
37      Nome,
38      Estado_Civil,
39      REPLACE(Estado_Civil, 'S', 'Solteiro') AS 'Estado_Civil'
40  FROM clientes;
41

```

	Nome	Estado_Civil	Estado_Civil
▶	Ruben	C	C
	Christy	S	Solteiro
	Elizabeth	S	Solteiro
	Julio	S	Solteiro
	Janet	S	Solteiro
	Marco	C	C
	Rob	S	Solteiro
	Shannon	S	Solteiro
	Jacquelyn	S	Solteiro
	Curtis	C	C
	Lauren	C	C
	Ian	C	C
	Sydney	S	Solteiro
	Chloe	S	Solteiro
	Wyatt	C	C
	Shannon	S	Solteiro
	Clarence	S	Solteiro

Função REPLACE

Como queremos fazer uma nova substituição (dessa vez 'C' por 'Casado') aplicamos a função REPLACE mais uma vez, e chegamos no resultado ao lado.

```

34  -- REPLACE
35 •  SELECT * FROM clientes;
36 •  SELECT
37      Nome,
38      Estado_Civil,
39      REPLACE(REPLACE(Estado_Civil, 'S', 'Solteiro'), 'C', 'Casado') AS 'Estado_Civil'
40  FROM clientes;
41

```

	Nome	Estado_Civil	Estado_Civil
▶	Ruben	C	Casado
	Christy	S	Solteiro
	Elizabeth	S	Solteiro
	Julio	S	Solteiro
	Janet	S	Solteiro
	Marco	C	Casado
	Rob	S	Solteiro
	Shannon	S	Solteiro
	Jacquelyn	S	Solteiro
	Curtis	C	Casado
	Lauren	C	Casado
	Ian	C	Casado
	Sydney	S	Solteiro
	Chloe	S	Solteiro
	Wyatt	C	Casado
	Shannon	S	Solteiro
	Clarence	S	Solteiro

Função INSTR

Outra função de texto muito útil é a INSTR.

Ela é capaz de retornar a posição que um determinado caractere ocupa dentro de um texto.

Por exemplo, no e-mail abaixo, a posição que o caractere '@' ocupa é a posição 8

r u b e n 3 5 @ h o t m a i l . c o m
1 2 3 4 5 6 7 8 9 . . .

Função INSTR

Aplicando essa função na coluna Email da tabela clientes, chegamos ao resultado para cada email.

```
42    -- INSTR  
43 • SELECT  
44      Email,  
45      INSTR(Email, '@') FROM clientes;
```

	Email	INSTR(Email, '@')
▶	ruben35@hotmail.com	8
	christy12@hotmail.com	10
	elizabeth5@gmail.com	11
	julio1@hotmail.com	7
	janet9@gmail.com	7
	marco14@yahoo.com.br	8
	rob4@hotmail.com	5
	shannon38@gmail.com	10

Função MID

Podemos combinar a função INSTR com uma nova função, chamada MID.

Essa função permite retornar uma sequência de caracteres, a partir dos seguintes argumentos:

- 1) Texto
- 2) Posição inicial
- 3) Quantidade de caracteres desejados a partir da posição inicial.

Por exemplo, se quisermos extrair o ID do e-mail (ou seja, tudo o que vem antes do @), podemos usar a função MID como mostrado.

Primeiro informamos o e-mail, em seguida informamos a posição inicial em que começa o texto que queremos extrair, e depois a quantidade de caracteres que queremos a partir da posição inicial. Como o texto ‘ruben35’ tem 7 caracteres, passamos esse valor dentro da função.



```

42      -- MID
43 •  SELECT Email,
44        MID('ruben35@hotmail.com', 1, 7)
45
46
47
48
49

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Email	MID('ruben35@hotmail.com', 1, 7)		
ruben35@hotmail.com	ruben35		

ruben35@hotmail.com

ID do
e-mail

Função MID e INSTR

Para tornar o processo de extrair a primeira parte dos e-mails mais automática, podemos combinar a função INSTR com a função MID.

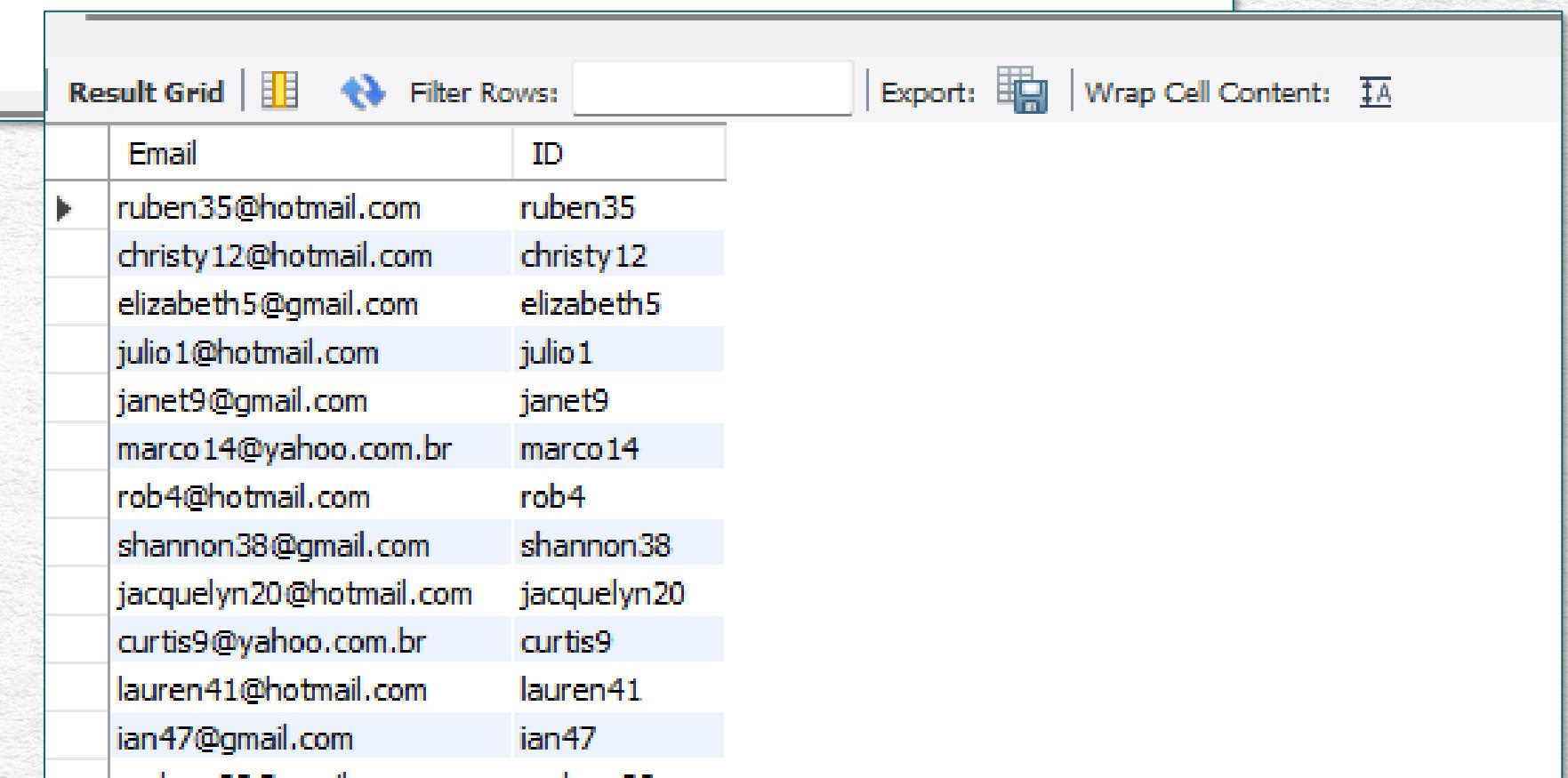
Observe que o tamanho do ID é sempre a posição do '@' em cada um dos e-mails.

Ao lado, temos a solução final.

```

47 • SELECT
48     Email,
49     MID(Email, 1, INSTR(Email, '@') - 1) 'ID'
50 FROM clientes;
51
52
53

```



	Email	ID
▶	ruben35@hotmail.com	ruben35
	christy12@hotmail.com	christy12
	elizabeth5@gmail.com	elizabeth5
	julio1@hotmail.com	julio1
	janet9@gmail.com	janet9
	marco14@yahoo.com.br	marco14
	rob4@hotmail.com	rob4
	shannon38@gmail.com	shannon38
	jacquelyn20@hotmail.com	jacquelyn20
	curtis9@yahoo.com.br	curtis9
	lauren41@hotmail.com	lauren41
	ian47@gmail.com	ian47

Funções de Data

Vejamos agora as funções de Data mais comuns.

STRINGS

Funções para trabalhar
com **TEXTOS**.

- LENGTH
- CONCAT
- CONCAT_WS
- LCASE e UCASE
- RIGHT e LEFT
- REPLACE
- INSTR
- MID

DATAS

Funções para trabalhar
com **DATAS**.

- DAY
- YEAR
- MONTH
- NOW
- CURDATE
- CURTIME
- DATE_DIFF
- DATE_ADD
- DATE_SUB

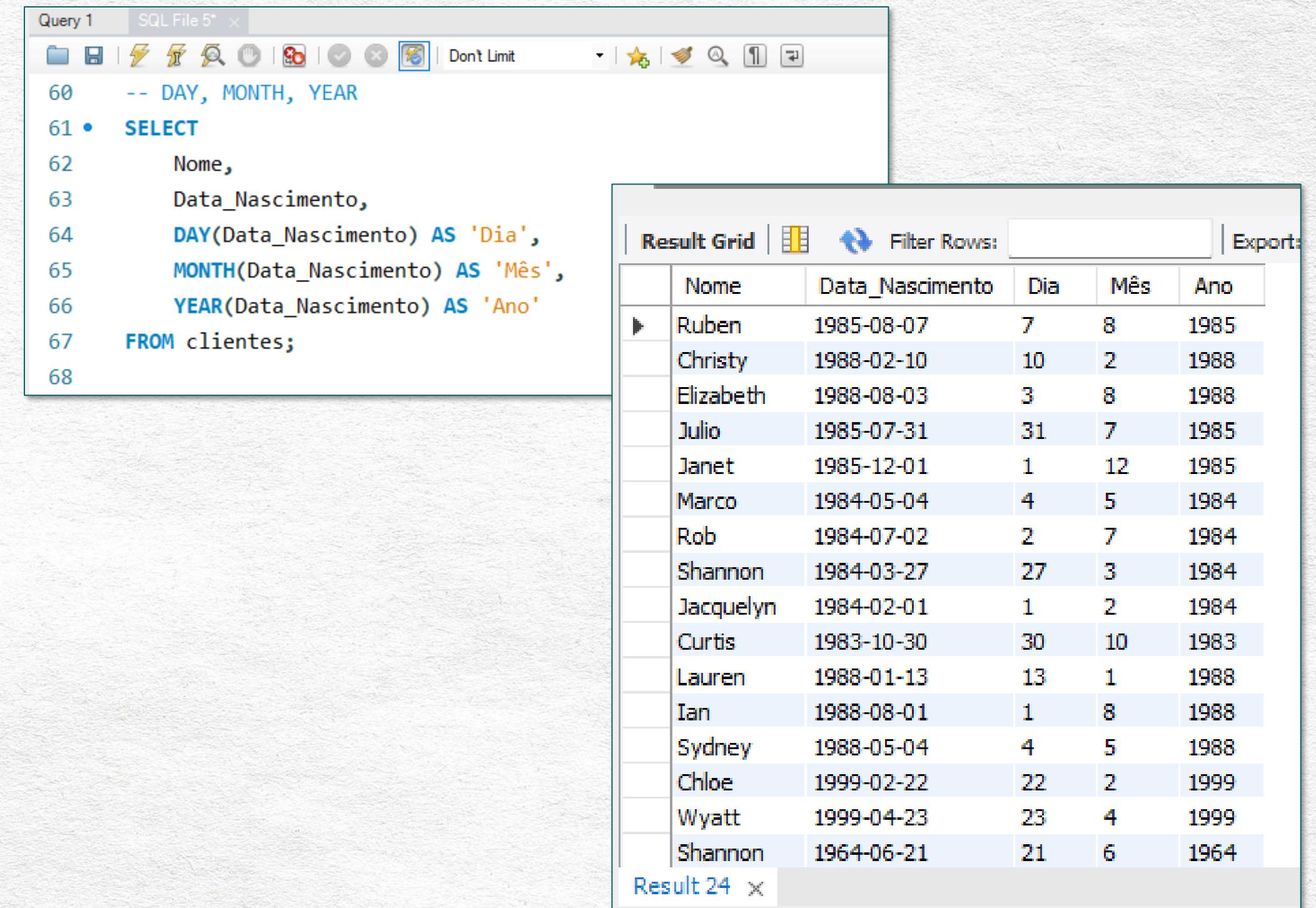


Função DAY, MONTH e YEAR

As primeiras funções de datas que veremos serão: DAY, MONTH e YEAR.

Essas funções permitem retornar, respectivamente, o dia, mês e ano de uma data.

Suas aplicações são bem diretas e mostradas ao lado, onde queremos saber qual é o dia, mês e ano de cada data de nascimento dos clientes.



The screenshot shows a MySQL Workbench interface. On the left, the 'Query 1' tab displays the following SQL code:

```

60  -- DAY, MONTH, YEAR
61 • SELECT
62   Nome,
63   Data_Nascimento,
64   DAY(Data_Nascimento) AS 'Dia',
65   MONTH(Data_Nascimento) AS 'Mês',
66   YEAR(Data_Nascimento) AS 'Ano'
67 FROM clientes;
68

```

On the right, the 'Result Grid' shows the output of the query. The results are presented in a table with the following columns: Nome, Data_Nascimento, Dia, Mês, and Ano. The data consists of 24 rows of client birth information.

	Nome	Data_Nascimento	Dia	Mês	Ano
▶	Ruben	1985-08-07	7	8	1985
	Christy	1988-02-10	10	2	1988
	Elizabeth	1988-08-03	3	8	1988
	Julio	1985-07-31	31	7	1985
	Janet	1985-12-01	1	12	1985
	Marco	1984-05-04	4	5	1984
	Rob	1984-07-02	2	7	1984
	Shannon	1984-03-27	27	3	1984
	Jacquelyn	1984-02-01	1	2	1984
	Curtis	1983-10-30	30	10	1983
	Lauren	1988-01-13	13	1	1988
	Ian	1988-08-01	1	8	1988
	Sydney	1988-05-04	4	5	1988
	Chloe	1999-02-22	22	2	1999
	Wyatt	1999-04-23	23	4	1999
	Shannon	1964-06-21	21	6	1964

Result 24 ×

Função DATEDIFF

A função DATEDIFF retorna a diferença entre duas datas.

No exemplo abaixo, calculamos o tempo de entrega (em dias) entre duas datas.

```
86 -- DATEDIFF:  
87 -- Exemplo: Calcular o tempo de entrega (em dias) de um determinado projeto que teve início no dia 10/04/2021  
     e terminou no dia 15/07/2021.  
88  
89 • SET @varDataInicio = '2021-04-10';  
90 • SET @varDataFim = '2021-07-15';  
91  
92 • SELECT DATEDIFF(@varDataFim, @varDataInicio);  
93  
94  
95  
96
```

Result Grid	
	Filter Rows:
DATEDIFF(@varDataFim, @varDataInicio)	
▶ 96	

Função DATE_ADD

A função DATE_ADD adiciona uma quantidade de dias, meses ou anos a uma determinada data.

No exemplo abaixo, adicionamos 10 dias (INTERVAL 10 DAY) à data de início.

Se quiséssemos adicionar 1 mês, por exemplo, faríamos INTERVAL 1 MONTH, e para 2 anos, por exemplo, faríamos INTERVAL 2 YEAR.

```
78 -- DATE_ADD
79 -- Exemplo: Um projeto deve ser entregue 10 dias após a data de início. Qual é a data final da entrega?
80
81 • SET @varDataInicio = '2021-04-10';
82
83 • SELECT DATE_ADD(@varDataInicio, INTERVAL 10 DAY);
84
85
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

DATE_ADD(@varDataInicio, INTERVAL 10 DAY)
2021-04-20

Função DATE_SUB

A função DATE_SUB subtrai uma quantidade de dias, meses ou anos de uma determinada data.

Observe o exemplo de aplicação abaixo:

```
--  
96  -- DATE_SUB  
97  -- Exemplo: Um projeto finalizou no dia 21/09/2021 e teve 10 dias de duração. Qual foi a data de início?  
98  
99 • SET @varDataFim = '2021-09-21';  
100  
101 • SELECT DATE_SUB(@varDataFim, INTERVAL 10 DAY);  
102  
103
```

Result Grid | Filter Rows: _____ | Export: _____ | Wrap Cell Content: _____

	DATE_SUB(@varDataFim, INTERVAL 10 DAY)
▶	2021-09-11

MÓDULO 9

JOINS

JOINS

JOINS



Chave Primária vs. Chave Estrangeira

Uma **Chave Primária** é uma coluna que identifica as informações distintas em uma tabela. Geralmente é uma coluna de ID. Toda tabela terá uma, e somente uma, chave primária. Essa chave é utilizada como identificador único da tabela, sendo representada por uma coluna que não receberá valores repetidos.

Chave Primária vs. Chave Estrangeira

Uma **Chave Primária** é uma coluna que identifica as informações distintas em uma tabela. Geralmente é uma coluna de ID. Toda tabela terá uma, e somente uma, chave primária. Essa chave é utilizada como identificador único da tabela, sendo representada por uma coluna que não receberá valores repetidos.

Como pode ser visto abaixo, a tabela **Produtos** possui uma coluna chamada **ID_Produto**, com valores que não se repetem. Essa será a **Chave Primária**.

Já na tabela de **Pedidos**, temos uma coluna chamada **ID_Pedido**, e essa também não se repete. Entendemos essa coluna também como uma chave primária.

produtos

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214.5
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160

pedidos

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	2	11	84	1	800	344	800	344
3	2019/01/01	1	14	12	1	3100	1209	3100	1209
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	5	8	21	1	600	258	600	258
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	850	3400	850
8	2019/01/01	3	3	73	1	450	90	450	90
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	1	4	27	1	350	129.5	350	129.5
11	2019/01/01	5	9	25	1	780	327.6	780	327.6
12	2019/01/01	5	4	75	1	350	129.5	350	129.5
13	2019/01/01	5	11	48	1	800	344	800	344
14	2019/01/01	5	16	8	1	2900	1160	2900	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3100	1209	3100	1209
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	65	1	450	90	450	90
20	2019/01/01	7	16	16	1	2900	1160	2900	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	36	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	129.5	350	129.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	7	1	77	1	2300	966	2300	966

Chave Primária vs. Chave Estrangeira

Uma **Chave Primária** é uma coluna que identifica as informações distintas em uma tabela. Geralmente é uma coluna de ID. Toda tabela terá uma, e somente uma, chave primária. Essa chave é utilizada como identificador único da tabela, sendo representada por uma coluna que não receberá valores repetidos.

Como pode ser visto abaixo, a tabela **Produtos** possui uma coluna chamada **ID_Produto**, com valores que não se repetem. Essa será a **Chave Primária**.

Já na tabela de **Pedidos**, temos uma coluna chamada **ID_Pedido**, e essa também não se repete. Entendemos essa coluna também como uma chave primária.

produtos

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214.5
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160

Chave Primária

pedidos

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	2	11	84	1	800	344	800	344
3	2019/01/01	1	14	12	1	3100	1209	3100	1209
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	5	8	21	1	600	258	600	258
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	850	3400	850
8	2019/01/01	3	3	73	1	450	90	450	90
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	1	4	27	1	350	129.5	350	129.5
11	2019/01/01	5	9	25	1	780	327.6	780	327.6
12	2019/01/01	5	4	75	1	350	129.5	350	129.5
13	2019/01/01	5	11	48	1	800	344	800	344
14	2019/01/01	5	16	8	1	2900	1160	2900	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3100	1209	3100	1209
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	65	1	450	90	450	90
20	2019/01/01	7	16	16	1	2900	1160	2900	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	36	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	129.5	350	129.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	7	1	77	1	2300	966	2300	966

Chave Primária



Chave Primária vs. Chave Estrangeira

Já uma **Chave Estrangeira** é uma coluna que permite relacionar as linhas de uma segunda tabela com a **Chave Primária** de uma primeira tabela.

produtos

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214.5
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160

pedidos

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	2	11	84	1	800	344	800	344
3	2019/01/01	1	14	12	1	3100	1209	3100	1209
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	5	8	21	1	600	258	600	258
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	850	3400	850
8	2019/01/01	3	3	73	1	450	90	450	90
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	1	4	27	1	350	129.5	350	129.5
11	2019/01/01	5	9	25	1	780	327.6	780	327.6
12	2019/01/01	5	4	75	1	350	129.5	350	129.5
13	2019/01/01	5	11	48	1	800	344	800	344
14	2019/01/01	5	16	8	1	2900	1160	2900	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3100	1209	3100	1209
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	65	1	450	90	450	90
20	2019/01/01	7	16	16	1	2900	1160	2900	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	36	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	129.5	350	129.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	7	1	77	1	2300	966	2300	966

Chave Primária vs. Chave Estrangeira

Já uma **Chave Estrangeira** é uma coluna que permite relacionar as linhas de uma segunda tabela com a **Chave Primária** de uma primeira tabela.

Como pode ser visto abaixo, a tabela **Produtos** possui uma coluna chamada **ID_Produto**, com valores que não se repetem. Essa será a **Chave Primária**. Já na tabela de **Pedidos**, a coluna de **ID_Produto** também aparece, mas os valores se repetem. Isso porque podemos ter mais de um pedido de um mesmo produto. Na tabela de **Pedidos**, a coluna de **ID_Produto** vai ser a **Chave Estrangeira** e vai permitir a gente relacionar os valores dessa coluna com a **Chave Primária** da tabela de **Produtos**.

produtos

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214.5
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160

pedidos

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	2	11	84	1	800	344	800	344
3	2019/01/01	1	14	12	1	3100	1209	3100	1209
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	5	8	21	1	600	258	600	258
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	850	3400	850
8	2019/01/01	3	3	73	1	450	90	450	90
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	1	4	27	1	350	129.5	350	129.5
11	2019/01/01	5	9	25	1	780	327.6	780	327.6
12	2019/01/01	5	4	75	1	350	129.5	350	129.5
13	2019/01/01	5	11	48	1	800	344	800	344
14	2019/01/01	5	16	8	1	2900	1160	2900	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3100	1209	3100	1209
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	65	1	450	90	450	90
20	2019/01/01	7	16	16	1	2900	1160	2900	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	36	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	129.5	350	129.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	7	1	77	1	2300	966	2300	966

Chave Primária vs. Chave Estrangeira

Já uma **Chave Estrangeira** é uma coluna que permite relacionar as linhas de uma segunda tabela com a **Chave Primária** de uma primeira tabela.

Como pode ser visto abaixo, a tabela **Produtos** possui uma coluna chamada **ID_Produto**, com valores que não se repetem. Essa será a **Chave Primária**. Já na tabela de **Pedidos**, a coluna de **ID_Produto** também aparece, mas os valores se repetem. Isso porque podemos ter mais de um pedido de um mesmo produto. Na tabela de **Pedidos**, a coluna de **ID_Produto** vai ser a **Chave Estrangeira** e vai permitir a gente relacionar os valores dessa coluna com a Chave Primária da tabela de **Produtos**.

produtos

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214.5
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160

Chave Primária

Chave Estrangeira

pedidos

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	2	11	84	1	800	344	800	344
3	2019/01/01	1	14	12	1	3100	1209	3100	1209
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	5	8	21	1	600	258	600	258
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	850	3400	850
8	2019/01/01	3	3	73	1	450	90	450	90
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	1	4	27	1	350	129.5	350	129.5
11	2019/01/01	5	9	25	1	780	327.6	780	327.6
12	2019/01/01	5	4	75	1	350	129.5	350	129.5
13	2019/01/01	5	11	48	1	800	344	800	344
14	2019/01/01	5	16	8	1	2900	1160	2900	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3100	1209	3100	1209
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	65	1	450	90	450	90
20	2019/01/01	7	16	16	1	2900	1160	2900	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	36	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	129.5	350	129.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	7	1	77	1	2300	966	2300	966

Tabela Fato vs. Tabela Dimensão

Uma **Tabela Dimensão** é uma tabela que contém características de um determinado elemento: lojas, produtos, funcionários, clientes, etc.

Nesta tabela, nenhum dos elementos principais irá se repetir. É onde vamos encontrar nossas **chaves primárias**.

Já uma **Tabela Fato** é uma tabela que vai registrar os fatos ou acontecimentos de uma empresa/negócio em determinados períodos de tempo (vendas, devoluções, aberturas de chamados, receitas, despesas, etc).

Geralmente é uma tabela com milhares de informações e composta essencialmente por colunas de ID usadas para buscar as informações complementares de uma tabela dimensão, conhecidas como **chaves estrangeiras**.

No exemplo ao lado, a FactSales é a nossa tabela Fato e a DimChannel é a nossa tabela Dimensão.

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214.5
13	notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160

produtos

Chave Primária

ID_Pedido	Data_Venda	ID_Linha	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019050801	8	33	45	1	800	344	800	344
2	2019050801	2	33	94	1	800	344	800	344
3	2019050801	1	34	52	1	3000	1200	3000	1200
4	2019050801	8	9	98	1	700	327.6	700	327.6
5	2019050801	9	8	21	1	600	258	600	258
6	2019050801	8	8	26	1	600	258	600	258
7	2019050801	6	13	90	1	3400	950	3400	950
8	2019050801	3	3	79	1	400	90	400	90
9	2019050801	5	15	17	1	4000	1176	4000	1176
10	2019050801	1	4	27	1	300	120.5	300	120.5
11	2019050801	9	9	25	1	700	327.6	700	327.6
12	2019050801	9	4	25	1	300	120.5	300	120.5
13	2019050801	9	11	48	1	600	344	600	344
14	2019050801	9	26	8	1	2000	1160	2000	1160
15	2019050801	9	9	41	1	700	327.6	700	327.6
16	2019050801	8	1	6	1	2000	966	2000	966
17	2019050801	6	14	77	1	3000	1200	3000	1200
18	2019050801	9	18	55	1	4000	1176	4000	1176
19	2019050801	9	3	45	1	400	90	400	90
20	2019050801	7	38	36	1	2000	1160	2000	1160
21	2019050801	9	11	55	1	600	344	600	344
22	2019050801	6	9	36	1	700	327.6	700	327.6
23	2019050801	3	4	42	1	300	120.5	300	120.5
24	2019050801	9	12	81	1	600	214.5	600	214.5

pedidos

Tabela Fato vs. Tabela Dimensão

Uma **Tabela Dimensão** é uma tabela que contém características de um determinado elemento: lojas, produtos, funcionários, clientes, etc.

Nesta tabela, nenhum dos elementos principais irá se repetir. É onde vamos encontrar nossas **chaves primárias**.

Já uma **Tabela Fato** é uma tabela que vai registrar os fatos ou acontecimentos de uma empresa/negócio em determinados períodos de tempo (vendas, devoluções, aberturas de chamados, receitas, despesas, etc).

Geralmente é uma tabela com milhares de informações e composta essencialmente por colunas de ID usadas para buscar as informações complementares de uma tabela dimensão, conhecidas como **chaves estrangeiras**.

No exemplo ao lado, a tabela Pedidos é a nossa tabela Fato e a Produtos é a nossa tabela Dimensão.

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Nome_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-CL-033823	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-OS-001839	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WEB-C7-034604	450	90
4	Kit Teclado + Mouse sem Fio Wireless	2	DELL	KTH-CL-043839	280	128.5
5	Kit Teclado + Mouse Sem Bluetooth	2	DELL	KTH-CL-111824	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGR-AL-000142	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGR-AL-014971	2100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-294621	600	258
9	Fone de Ouvido Tune T5000	4	BEATS	HDP-BE-091834	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237991	1200	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-019403	800	344
12	Microfone de mesa com Fio condensador	5	BLUE	MIC-BL-761411	650	214.5
13	Netbook LC2100 Intel Core i5 8GB	6	SAMSUNG	NET-OS-918457	3400	850
14	Netbook Inspiron 15 5000 4GB	6	DELL	NET-CL-000012	3100	1209
15	Netbook IdeaPad RF32000	6	DELL	NET-CL-773846	4200	1176
16	Netbook Holden Ultra 2	6	SAMSUNG	NET-OS-12138U	2900	1180

produtos

Chave Estrangeira

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	2	11	84	1	800	344	800	344
3	2019/01/01	1	14	12	1	3100	1209	3100	1209
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	5	8	21	1	600	258	600	258
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	850	3400	850
8	2019/01/01	3	3	73	1	450	90	450	90
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	1	4	27	1	350	129.5	350	129.5
11	2019/01/01	5	9	25	1	780	327.6	780	327.6
12	2019/01/01	5	4	75	1	350	129.5	350	129.5
13	2019/01/01	5	11	48	1	800	344	800	344
14	2019/01/01	5	16	8	1	2900	1160	2900	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3100	1209	3100	1209
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	65	1	450	90	450	90
20	2019/01/01	7	16	16	1	2900	1160	2900	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	36	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	129.5	350	129.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	7	1	77	1	2300	966	2300	966

pedidos

Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

produtos

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214.5
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SS-918457	3400	850
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOT-DL-77164I	4200	1176
16	Notebook Motion Ultra 2	6	SAMSUNG	NOT-SS-13139U	2900	1160

Chave Primária

Tabela Dimensão

Chave Estrangeira

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	2	11	84	1	800	344	800	344
3	2019/01/01	1	14	12	1	3100	1209	3100	1209
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	5	8	21	1	600	258	600	258
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	850	3400	850
8	2019/01/01	3	3	73	1	450	90	450	90
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	1	4	27	1	350	129.5	350	129.5
11	2019/01/01	5	9	25	1	780	327.6	780	327.6
12	2019/01/01	5	4	75	1	350	129.5	350	129.5
13	2019/01/01	5	11	48	1	800	344	800	344
14	2019/01/01	5	16	8	1	2900	1160	2900	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3100	1209	3100	1209
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	65	1	450	90	450	90
20	2019/01/01	7	16	16	1	2900	1160	2900	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	36	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	129.5	350	129.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	7	1	77	1	2300	966	2300	966

Tabela Fato

Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

Mas pra que servem essas relações?

Com essas relações, conseguimos utilizar informações de uma tabela em outra tabela. Será muito útil, por exemplo, pra gente descobrir o nome do produto vendido (na tabela de Pedidos) fazendo essa busca lá na tabela de Produtos.

Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

Mas pra que servem essas relações?

Com essas relações, conseguimos utilizar informações de uma tabela em outra tabela. Será muito útil, por exemplo, pra gente descobrir o nome do produto vendido (na tabela de Pedidos) fazendo essa busca lá na tabela de Produtos.

pedidos

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	3	14	12	1	3000	3299	3000	3299
4	2019/01/01	8	9	98	1	700	327.6	700	327.6
5	2019/01/01	9	8	21	1	600	299	600	299
6	2019/01/01	6	8	26	1	600	258	600	258
7	2019/01/01	6	13	90	1	3400	890	3400	890
8	2019/01/01	3	3	79	1	400	96	400	96
9	2019/01/01	5	15	17	1	4200	1176	4200	1176
10	2019/01/01	3	4	27	1	300	329.5	300	329.5
11	2019/01/01	9	9	25	1	700	327.6	700	327.6
12	2019/01/01	6	4	79	1	300	329.5	300	329.5
13	2019/01/01	9	11	40	1	800	344	800	344
14	2019/01/01	6	18	8	1	2800	1160	2800	1160
15	2019/01/01	5	9	41	1	700	327.6	700	327.6
16	2019/01/01	8	1	6	1	2300	966	2300	966
17	2019/01/01	6	14	77	1	3000	3299	3000	3299
18	2019/01/01	5	15	55	1	4200	1176	4200	1176
19	2019/01/01	5	3	63	1	400	96	400	96
20	2019/01/01	7	16	38	1	2800	1160	2800	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	38	1	700	327.6	700	327.6
23	2019/01/01	3	4	42	1	300	329.5	300	329.5
24	2019/01/01	5	12	81	1	600	214.5	600	214.5
25	2019/01/01	6	1	11	1	11000	866	11000	866



Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

Mas pra que servem essas relações?

Com essas relações, conseguimos utilizar informações de uma tabela em outra tabela. Será muito útil, por exemplo, pra gente descobrir o nome do produto vendido (na tabela de Pedidos) fazendo essa busca lá na tabela de Produtos.

produtos

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MONIT-01-019403	2300	964
2	Monitor Curvo 24" 144Hz HDMI	3	SAMSUNG	MONIT-02-019399	2800	980
3	Webcam Full HD 1080p	1	LOGITECH	WEB-01-014604	450	90
4	Kit Teclado + Mouse sem Fio Wireless	2	DELL	KIT-01-010399	250	125.5
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KIT-01-111924	280	139.2
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CDH-AL-000142	1800	940
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CDH-AL-014070	2100	1395
8	Headphone Bluetooth 2000	4	SONY	HDP-SB-194821	600	298
9	Fone de Ouvido TWS T5000	4	BL	HDP-BB-019234	780	327.6
10	Microfone Condensador MC2000	5	AKG	MIC-AK-217991	1100	215
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344
12	Microfone de mesa com fio condensador	5	BLUE	MIC-BL-7819433	650	214.5
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOTE-03-918457	3400	890
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOTE-01-000012	3100	1209
15	Notebook IdeaPad RF32000	6	DELL	NOTE-01-771646	4200	1176
16	Notebook Moltar Ultra 2	6	SAMSUNG	NOTE-01-521396U	2900	1360

pedidos

ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Venda	Preco_Unit	Custo_Unit	Receita_Venda	Custo_Venda
1	2019/01/01	6	11	45	1	800	344	800	344
2	2019/01/01	3	14	12	1	3000	1200	3000	1200
4	2019/01/01	8	9	98	1	780	327.6	780	327.6
5	2019/01/01	9	8	21	1	600	256	600	256
6	2019/01/01	6	8	26	1	600	256	600	256
7	2019/01/01	6	13	90	1	3400	1360	3400	1360
8	2019/01/01	3	3	79	1	400	90	400	90
9	2019/01/01	5	15	17	1	4200	1376	4200	1376
10	2019/01/01	3	4	27	1	350	128.5	350	128.5
11	2019/01/01	9	9	25	1	780	327.6	780	327.6
12	2019/01/01	6	4	79	1	350	128.5	350	128.5
13	2019/01/01	9	11	40	1	800	344	800	344
14	2019/01/01	6	18	8	1	2800	1160	2800	1160
15	2019/01/01	5	9	41	1	780	327.6	780	327.6
16	2019/01/01	9	1	6	1	2300	960	2300	960
17	2019/01/01	6	14	77	1	3000	1200	3000	1200
18	2019/01/01	5	15	55	1	4200	1376	4200	1376
19	2019/01/01	5	3	63	1	400	90	400	90
20	2019/01/01	7	18	38	1	2800	1160	2800	1160
21	2019/01/01	5	11	55	1	800	344	800	344
22	2019/01/01	6	9	38	1	780	327.6	780	327.6
23	2019/01/01	3	4	42	1	350	128.5	350	128.5
24	2019/01/01	5	12	81	1	650	214.5	650	214.5
25	2019/01/01	9	1	111	1	11000	4400	11000	4400

Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

Mas pra que servem essas relações?

Com essas relações, conseguimos utilizar informações de uma tabela em outra tabela. Será muito útil, por exemplo, pra gente descobrir o nome do produto vendido (na tabela de Pedidos) fazendo essa busca lá na tabela de Produtos.

Essas relações serão criadas por meio do que chamamos de JOIN's. A tradução literal dessa palavra é “juntar”, “unir”. Os JOINs vão nos permitir fazer exatamente isso: juntar as nossas tabelas Fato e Dimensão, de forma a complementar as informações umas das outras.

Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

Mas pra que servem essas relações?

Com essas relações, conseguimos utilizar informações de uma tabela em outra tabela. Será muito útil, por exemplo, pra gente descobrir o nome do produto vendido (na tabela de Pedidos) fazendo essa busca lá na tabela de Produtos.

Essas relações serão criadas por meio do que chamamos de JOIN's. A tradução literal dessa palavra é “juntar”, “unir”. Os JOINs vão nos permitir fazer exatamente isso: juntar as nossas tabelas Fato e Dimensão, de forma a complementar as informações umas das outras.

Existem diversos tipos de Joins no SQL, os quais estão listados ao lado.

LEFT JOIN

RIGHT JOIN

INNER JOIN

FULL JOIN

Nas aulas anteriores vimos que existem dois tipos de tabelas: Dimensão e Fato. Essas tabelas podem ser relacionadas através de uma coluna: na tabela Dimensão, identificamos a chave Primária, que será relacionada com a chave Estrangeira da tabela Fato.

Mas pra que servem essas relações?

Com essas relações, conseguimos utilizar informações de uma tabela em outra tabela. Será muito útil, por exemplo, pra gente descobrir o nome do produto vendido (na tabela de Pedidos) fazendo essa busca lá na tabela de Produtos.

Essas relações serão criadas por meio do que chamamos de JOIN's. A tradução literal dessa palavra é “juntar”, “unir”. Os JOINs vão nos permitir fazer exatamente isso: juntar as nossas tabelas Fato e Dimensão, de forma a complementar as informações umas das outras.

Existem diversos tipos de Joins no SQL, os quais estão listados ao lado.

LEFT JOIN

RIGHT JOIN

INNER JOIN

FULL JOIN

INNER JOIN

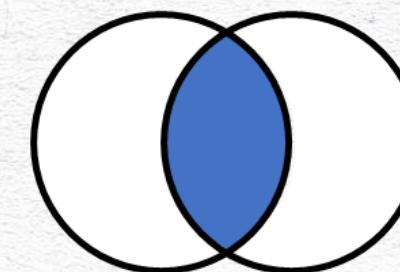
pedidos

ID_Produto	Data_Venda	Qtd_Vendida	Receita_Venda
3	2019/01/01	1	450
4	2019/01/01	1	350
4	2019/01/01	1	350
1	2019/01/01	1	2300
3	2019/01/01	1	450
4	2019/01/01	1	350
1	2019/01/01	1	2300
2	2019/01/01	1	2800
4	2019/01/01	1	350
1	2019/01/01	1	2300

Apenas a interseção entre Tabelas A e B

Tabela A

Tabela B



Resultado

produtos

ID_Produto	Nome_Produto	Marca_Produto	Preco_Unit
1	Monitor LED 19,5" Full HD HDMI	DELL	2300
2	Monitor Curvo 24" 144Hz HDMI	SAMSUNG	2800
3	Webcam Full HD 1080p	LOGITECH	450
4	Kit Teclado + Mouse sem fio Wireless	DELL	350

ID_Produto	Data_Venda	Qtd_Vendida	Receita_Venda	Nome_Produto	Marca_Produto	Preco_Unit
3	2019/01/01	1	450	Webcam Full HD 1080p	LOGITECH	450
4	2019/01/01	1	350	Kit Teclado + Mouse sem fio Wireless	DELL	350
4	2019/01/01	1	350	Kit Teclado + Mouse sem fio Wireless	DELL	350
1	2019/01/01	1	2300	Monitor LED 19,5" Full HD HDMI	DELL	2300
3	2019/01/01	1	450	Webcam Full HD 1080p	LOGITECH	450
4	2019/01/01	1	350	Kit Teclado + Mouse sem fio Wireless	DELL	350
1	2019/01/01	1	2300	Monitor LED 19,5" Full HD HDMI	DELL	2300
2	2019/01/01	1	2800	Monitor Curvo 24" 144Hz HDMI	SAMSUNG	2800
4	2019/01/01	1	350	Kit Teclado + Mouse sem fio Wireless	DELL	350
1	2019/01/01	1	2300	Monitor LED 19,5" Full HD HDMI	DELL	2300

O INNER JOIN vai nos permitir relacionar duas tabelas (Tabela A e Tabela B) e criar uma nova tabela (Tabela C) que é a junção das duas.

A tabela resultante desse JOIN terá **apenas as linhas que são a interseção entre a Tabela A e a Tabela B**.

INNER JOIN: Estrutura



SELECT

Tabela_A.coluna1,
Tabela_A.coluna2,
Tabela_A.coluna3,
Tabela_B.coluna4

FROM

Tabela_A

INNER JOIN Tabela_B

ON Tabela_A.id_chave_estrangeira = Tabela_B.id_chave_primaria

A Tabela_A será a tabela que vamos querer complementar com as informações da Tabela_B.

INNER JOIN (Exemplo 1)

Agora vamos ver na prática como se dá a relação entre as tabelas do banco de dados do curso.

Vamos começar relacionando as tabelas produtos e pedidos.

Isso só é possível pois essas duas tabelas possuem uma coluna em comum: a coluna **ID_Produto**.

```

104    -- Exemplo 1. Relacione as tabelas de Produtos e Pedidos.
105
106 • SELECT * FROM produtos;
107 • SELECT * FROM pedidos;
108
  
```

ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit			
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966			
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980			
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90			
4	Kit Teclado + Mouse sem Fio Wireless	2	DELL	KTM-DL-041039	350	129.5			
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2			
ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Receita_Venda	Custo_Venda	Custo_Unit	Preco_Unit
1	2019-01-01	2	4	16	1	350	129.5	129.5	350
2	2019-01-01	1	4	17	1	350	129.5	129.5	350
3	2019-01-01	3	4	23	1	350	129.5	129.5	350
4	2019-01-01	8	4	24	1	350	129.5	129.5	350
5	2019-01-01	4	4	25	1	350	129.5	129.5	350
6	2019-01-01	4	4	26	1	350	129.5	129.5	350
7	2019-01-01	2	6	14	1	1800	540	540	1800
8	2019-01-01	8	6	15	1	1800	540	540	1800
9	2019-01-01	7	6	38	1	1800	540	540	1800
10	2019-01-02	3	3	27	1	450	90	90	450
11	2019-01-02	4	3	28	1	450	90	90	450
12	2019-01-02	8	3	29	1	450	90	90	450
13	2019-01-02	2	3	30	1	450	90	90	450
14	2019-01-02	4	3	31	1	450	90	90	450
15	2019-01-02	7	3	32	1	450	90	90	450

INNER JOIN (Exemplo 1)

Seguindo o que foi explicado no início do módulo, podemos criar a relação entre as duas tabelas através da coluna ID_Produto conforme mostrado na imagem ao lado.

Lembrando que a coluna ID_Produto é uma chave primária da tabela Produtos, e uma chave estrangeira da tabela Pedidos.

Além disso, temos como tabela A a nossa tabela fato de Pedidos, enquanto a tabela B, dimensão, será a tabela de Produtos.

Observe agora que conseguimos visualizar na tabela de pedidos as colunas provenientes da tabela de produtos.

```

109 • SELECT
110     pedidos.ID_Pedido,
111     pedidos.Data_Venda,
112     pedidos.ID_Produto,
113     pedidos.Qtd_Vendida,
114     pedidos.Receita_Venda,
115     produtos.Nome_Produto,
116     produtos.Marca_Produto
117 FROM pedidos
118 INNER JOIN produtos
119     ON pedidos.ID_Produto = produtos.ID_Produto;
120

```

	ID_Pedido	Data_Venda	ID_Produto	Qtd_Vendida	Receita_Venda	Nome_Produto	Marca_Produto
▶	1	2019-01-01	4	1	350	Kit Tedado + Mouse sem fio Wireless	DELL
	2	2019-01-01	4	1	350	Kit Tedado + Mouse sem fio Wireless	DELL
	3	2019-01-01	4	1	350	Kit Tedado + Mouse sem fio Wireless	DELL
	4	2019-01-01	4	1	350	Kit Tedado + Mouse sem fio Wireless	DELL
	5	2019-01-01	4	1	350	Kit Tedado + Mouse sem fio Wireless	DELL
	6	2019-01-01	4	1	350	Kit Tedado + Mouse sem fio Wireless	DELL
	7	2019-01-01	6	1	1800	Cadeira Gamer reclinável Azul/Laranja	ALTURA
	8	2019-01-01	6	1	1800	Cadeira Gamer reclinável Azul/Laranja	ALTURA
	9	2019-01-01	6	1	1800	Cadeira Gamer reclinável Azul/Laranja	ALTURA
	10	2019-01-02	3	1	450	Webcam Full HD 1080p	LOGITECH
	11	2019-01-02	3	1	450	Webcam Full HD 1080p	LOGITECH

INNER JOIN (Exemplo 2)

Agora vamos relacionar as tabelas Clientes e Pedidos. Sabemos que as duas tabelas possuem uma coluna em comum: a coluna ID_Cliente.

Através dela, será possível criar a relação entre as tabelas.

```

104    -- Exemplo 1. Relacione as tabelas de Produtos e Pedidos.
105
106 • SELECT * FROM produtos;
107 • SELECT * FROM pedidos;
108

```

ID_Cliente	Nome	Sobrenome	Data_Nascimento	Estado_Civil	Sexo	Email	Telefone	Renda_Annual	Qtd_Filhos	Escolaridade
1	Ruben	Torres	1985-08-07	C	M	ruben35@hotmail.com	(85) 94132-1176	60000	3	Pós-graduado
2	Christy	Zhu	1988-02-10	S	F	christy12@hotmail.com	(41) 96270-6086	70000	0	Pós-graduado
3	Elizabeth	Johnson	1980-08-03	S	F	elizabeth5@gmail.com	(31) 92039-5832	80000	5	Pós-graduado
4	Juio	Ruiz	1985-07-31	S	M	julio1@hotmail.com	(62) 93391-5891	70000	0	Pós-graduado
5	Janet	Alvarez	1985-12-01	S	F	janet9@gmail.com	(21) 93379-3743	70000	0	Pós-graduado
6	Marco	Mehta	1984-05-04	C	M	marco14@yahoo.com.br	(85) 99019-3803	60000	3	Pós-graduado
7	Rob	Verhoff	1984-07-02	S		rob4@hotmail.com	(11) 98162-4760	60000	4	Pós-graduado
8	Shannon									
9	Jacquely									
10	Curtis									
11	Lauren									
ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Receita_Venda	Custo_Venda	Custo_Unit	Preco_Unit	
1	2019-01-01	2	4	16	1	350	129.5	129.5	350	
2	2019-01-01	1	4	17	1	350	129.5	129.5	350	
3	2019-01-01	3	4	23	1	350	129.5	129.5	350	
4	2019-01-01	8	4	24	1	350	129.5	129.5	350	
5	2019-01-01	4	4	25	1	350	129.5	129.5	350	
6	2019-01-01	4	4	26	1	350	129.5	129.5	350	
7	2019-01-01	2	6	14	1	1800	540	540	1800	
8	2019-01-01	8	6	15	1	1800	540	540	1800	
9	2019-01-01	7	6	38	1	1800	540	540	1800	
10	2019-01-02	3	3	27	1	450	90	90	450	
11	2019-01-02	4	3	28	1	450	90	90	450	
12	2019-01-02	8	3	29	1	450	90	90	450	
13	2019-01-02	2	3	30	1	450	90	90	450	
14	2019-01-02	4	3	31	1	450	90	90	450	
15	2019-01-02	7	3	32	1	450	90	90	450	

INNER JOIN (Exemplo 2)

Seguindo a mesma lógica do Exemplo 1, o código final que permite relacionar as tabelas de Clientes e Pedidos é mostrada ao lado.

```

127 • SELECT
128     pedidos.ID_Pedido,
129     pedidos.Data_Venda,
130     pedidos.ID_Cliente,
131     pedidos.Qtd_Vendida,
132     clientes.Nome,
133     clientes.Email,
134     clientes.Telefone
135 FROM pedidos
136 INNER JOIN clientes
137     ON pedidos.ID_Cliente = clientes.ID_Cliente;
138

```

	ID_Pedido	Data_Venda	ID_Cliente	Qtd_Vendida	Nome	Email	Telefone
▶	1	2019-01-01	16	1	Shannon	shannon1@hotmail.com	(71) 94496-1712
	2	2019-01-01	17	1	Clarence	clarence32@yahoo.com.br	(21) 98923-7805
	3	2019-01-01	23	1	Alejandro	alejandro45@gmail.com	(85) 94492-9582
	4	2019-01-01	24	1	Harold	harold3@hotmail.com	(31) 95052-6286
	5	2019-01-01	25	1	Jessie	jessie16@hotmail.com	(41) 98751-8234
	6	2019-01-01	26	1	Jill	jill13@gmail.com	(41) 97120-5603
	7	2019-01-01	14	1	Chloe	chloe23@hotmail.com	(71) 96890-7150
	8	2019-01-01	15	1	Wyatt	wyatt32@gmail.com	(71) 98644-2695
	9	2019-01-01	38	1	Jesse	jesse15@yahoo.com.br	(62) 99674-9226
	10	2019-01-02	27	1	Jimmy	jimmy9@yahoo.com.br	(85) 92931-9277
	11	2019-01-02	28	1	Bethany	bethany10@hotmail.com	(62) 92998-6359

Result 35 ×

Cuidados ao fazer os Joins

Será que somos obrigados a incluir o nome das tabelas antes das colunas?

Observe o código da situação 1 ao lado.
Selecionamos as colunas:

- ID_Pedido
- Data_Venda
- Qtd_Vendida
- Nome
- Email
- Telefone

Como pode ser visto, não informamos os nomes das tabelas e não tivemos nenhum problema. Mas será que é sempre assim?

127 • SELECT
 128 ID_Pedido,
 129 Data_Venda,
 130 Qtd_Vendida,
 131 Nome,
 132 Email,
 133 Telefone
 134 FROM pedidos
 135 INNER JOIN clientes
 136 ON pedidos.ID_Cliente = clientes.ID_Cliente;
 137

	ID_Pedido	Data_Venda	Qtd_Vendida	Nome	Email	Telefone
▶	1	2019-01-01	1	Shannon	shannon1@hotmail.com	(71) 94496-1712
	2	2019-01-01	1	Clarence	darence32@yahoo.com.br	(21) 98923-7805
	3	2019-01-01	1	Alejandro	alejandro45@gmail.com	(85) 94492-9582
	4	2019-01-01	1	Harold	harold3@hotmail.com	(31) 95052-6286
	5	2019-01-01	1	Jessie	jessie16@hotmail.com	(41) 98751-8234
	6	2019-01-01	1	Till	till13@hotmail.com	(41) 97120-5603

1

Cuidados ao fazer os Joins

Nessa nova situação, observe que incluímos a coluna ID_Cliente.

Ao executar o código, obtivemos o erro abaixo:

“Column ‘ID_Cliente’ in field list is ambiguous”

Mas o que significa esse erro?

Traduzindo, ele nos diz que o nome ID_Cliente é ambíguo. Isso porque essa coluna existe tanto na tabela pedidos quanto na tabela clientes. Dessa forma, o SQL não sabe de qual dessas duas tabelas selecionar a coluna ID_Cliente.

```

127 • SELECT
128     ID_Pedido,
129     Data_Venda,
130     ID_Cliente, ID_Cliente, ID_Cliente
131     Qtd_Vendida,
132     Nome,
133     Email,
134     Telefone
135 FROM pedidos
136 INNER JOIN clientes
137     ON pedidos.ID_Cliente = clientes.ID_Cliente;
138

```

2

 51 19:11:36 SELECT ID_Pedido, Data_Venda, ID_Cliente, Qtd_Vendida, Nome, Email, ... Error Code: 1052. Column 'ID_Cliente' in field list is ambiguous

Cuidados ao fazer os Joins

Para evitar este problema, podemos simplesmente colocar o nome da tabela antes de cada coluna, como pode ser visto ao lado.

Na situação 1, não tivemos problema porque nenhuma das colunas selecionadas existia nas duas tabelas (pedidos e clientes). Por isso não tivemos nenhum erro, pois não havia nenhuma coluna com nome ambíguo.

Já na situação 2, a coluna ID_Cliente existia tanto na tabela pedidos quanto na clientes, por isso o SQL não soube exatamente em qual das duas tabelas buscar, e deu esse erro.

A partir do momento que utilizamos o nome das tabelas, fica mais claro para o próprio SQL.

Portanto, o nome das tabelas não é obrigatório, mas acaba sendo o mais garantido a se fazer.

```
127 • SELECT
128     pedidos.ID_Pedido,
129     pedidos.Data_Venda,
130     pedidos.Qtd_Vendida,
131     clientes.ID_Cliente,
132     clientes.Nome,
133     clientes.Email,
134     clientes.Telefone
135 FROM pedidos
136 INNER JOIN clientes
137     ON pedidos.ID_Cliente = clientes.ID_Cliente;
```

	ID_Pedido	Data_Venda	Qtd_Vendida	ID_Cliente	Nome	Email	Telefone
▶	1	2019-01-01	1	16	Shannon	shannon1@hotmail.com	(71) 94496-1712
	2	2019-01-01	1	17	Clarence	clarence32@yahoo.com.br	(21) 98923-7805
	3	2019-01-01	1	23	Alejandro	alejandro45@gmail.com	(85) 94492-9582
	4	2019-01-01	1	24	Harold	harold3@hotmail.com	(31) 95052-6286
	5	2019-01-01	1	25	Jessie	jessie16@hotmail.com	(41) 98751-8234
	6	2019-01-01	1	26	Till	till13@gmail.com	(41) 97120-5603

Renomeando as tabelas com o ALIAS

Podemos dar um apelido para as tabelas com o objetivo de deixar o nosso código mais enxuto.

Observe no código ao lado. Utilizamos o AS para dar o nome 'p' à tabela pedidos e 'c' à tabela clientes. Em seguida, usamos esses novos nomes na hora de selecionar as colunas e também na hora de realizar o Join.

Em termos de resultado, temos exatamente a mesma coisa. A única diferença é que com o ALIAS de tabelas, nosso código fica bem mais resumido.

```

127 • SELECT
128     p.ID_Pedido,
129     p.Data_Venda,
130     p.Qtd_Vendida,
131     c.ID_Cliente,
132     c.Nome,
133     c.Email,
134     c.Telefone
135 FROM pedidos AS p
136 INNER JOIN clientes AS c
137     ON p.ID_Cliente = c.ID_Cliente;

```

	ID_Pedido	Data_Venda	Qtd_Vendida	ID_Cliente	Nome	Email	Telefone
▶	1	2019-01-01	1	16	Shannon	shannon1@hotmail.com	(71) 94496-1712
	2	2019-01-01	1	17	Clarence	darence32@yahoo.com.br	(21) 98923-7805
	3	2019-01-01	1	23	Alejandro	alejandro45@gmail.com	(85) 94492-9582
	4	2019-01-01	1	24	Harold	harold3@hotmail.com	(31) 95052-6286
	5	2019-01-01	1	25	Jessie	jessie16@hotmail.com	(41) 98751-8234
	6	2019-01-01	1	26	Till	till13@hotmail.com	(41) 97120-5603

Result 39 ×

INNER JOIN + WHERE

Os Joins podem ser combinados com o WHERE para filtrar a tabela.

No exemplo ao lado, incluímos no SELECT a coluna de sexo e fizemos um filtro WHERE para mostrar apenas os clientes do sexo masculino (WHERE c.Sexo = 'M').

Lembrando que a letra 'c' indica o nome da tabela clientes, tabela onde temos a coluna Sexo.

O WHERE é aplicado logo depois do Join pois é apenas depois do Join que temos a visibilidade da coluna Sexo em nossa consulta.

```

127 -- Seleciona todas os pedidos feitos pelos clientes do sexo masculino.
128
129 • SELECT
130     p.ID_Pedido,
131     p.Data_Venda,
132     p.Qtd_Vendida,
133     c.ID_Cliente,
134     c.Nome,
135     c.Email,
136     c.Telefone,
137     c.Sexo
138     FROM pedidos AS p
139     INNER JOIN clientes AS c
140         ON p.ID_Cliente = c.ID_Cliente
141     WHERE c.Sexo = 'M';
142

```

	ID_Pedido	Data_Venda	Qtd_Vendida	ID_Cliente	Nome	Email	Telefone	Sexo
▶	2	2019-01-01	1	17	Clarence	clarence32@yahoo.com.br	(21) 98923-7805	M
	3	2019-01-01	1	23	Alejandro	alejandro45@gmail.com	(85) 94492-9582	M
	4	2019-01-01	1	24	Harold	harold3@hotmail.com	(31) 95052-6286	M
	5	2019-01-01	1	25	Jessie	jessie16@hotmail.com	(41) 98751-8234	M
	8	2019-01-01	1	15	Wyatt	wyatt32@gmail.com	(71) 98644-2695	M
	9	2019-01-01	1	38	Jesse	jesse15@yahoo.com.br	(62) 99674-9226	M
	10	2019-01-02	1	27	Jimmy	jimmy9@yahoo.com.br	(85) 92931-9277	M
	14	2019-01-02	1	31	Jaime	jaime41@yahoo.com.br	(21) 96759-5499	M
	18	2019-01-03	1	37	Marc	marc3@yahoo.com.br	(11) 92792-7732	M

INNER JOIN + GROUP BY

Agora vamos ver como utilizar Joins com o Group By.

No exemplo ao lado, fizemos um agrupamento de ID_Loja na tabela pedidos, para mostrar a Receita Total e o Custo Total para cada loja.

Poderíamos melhorar essa consulta e mostrar nesse agrupamento, ao invés do ID da Loja, o seu respectivo nome.

```

143 -- Faça um agrupamento para mostrar o total de receita e custo por ID da loja.
144
145 • SELECT
146     ID_Loja,
147     SUM(Receita_Venda) AS 'Receita Total',
148     SUM(Custo_Venda) AS 'Custo Total'
149     FROM pedidos
150     GROUP BY ID_Loja;
151

```

	ID_Loja	Receita Total	Custo Total
▶	2	34400	14199.5
	1	30100	12901
	3	27800	11595.5
	8	27650	11468
	4	28600	11815
	7	17250	7080
	6	34300	14707
	5	28800	12150

INNER JOIN + GROUP BY

Para isso, vamos trocar dentro do SELECT a coluna ID_Loja por Loja.

Em seguida, é necessário fazer o INNER JOIN para que o código saiba onde buscar a coluna Loja.

Por fim, no GROUP BY, especificamos o nome da coluna a ser agrupado, que no caso, será Loja.

O resultado final é mostrado ao lado.

```

143 -- Faça um agrupamento para mostrar o total de receita e custo por Nome da loja.
144
145 • SELECT
146     Loja,
147     SUM(Receita_Venda) AS 'Receita Total',
148     SUM(Custo_Venda) AS 'Custo Total'
149     FROM pedidos
150     INNER JOIN lojas
151         ON pedidos.ID_Loja = lojas.ID_Loja
152     GROUP BY Loja;
153

```

	Loja	Receita Total	Custo Total
▶	Belo Horizonte	34400	14199.5
	Rio de Janeiro	30100	12901
	Salvador	27800	11595.5
	Goiânia	27650	11468
	Curitiba	28600	11815
	Niterói	17250	7080
	São Paulo	34300	14707
	Fortaleza	28800	12150

Result 42 ×

MÓDULO 10

FUNÇÕES CONDICIONAIS

FUNÇÕES CONDICIONAIS

FUNÇÕES CONDICIONAIS

Funções Condicionais

As funções condicionais nos permitem tratar condições dentro do MySQL. As funções condicionais que temos são:

IF

IFNULL

ISNULL

NULLIF

CASE

A seguir, vamos ver na prática como que cada uma delas vai funcionar.

A função IF permite tratar condições no MySQL. Para usá-la, precisamos informar 3 argumentos:

IF(teste_logico, valor_se_verdadeiro, valor_se_falso)

1

2

3

O **teste_logico** nada mais é do que a comparação entre dois valores. Por exemplo, $10 > 20$ (ou seja, o valor 10 é maior que o valor 20?).

Se a comparação do **teste_logico** for verdadeira, o resultado será o especificado no segundo argumento, **valor_se_verdadeiro**.

Se a comparação do **teste_logico** for falsa, o resultado será o especificado no terceiro argumento, **valor_se_falso**.

IF (Exemplo 1)

Observe o exemplo ao lado.

Temos uma comparação simples, onde queremos verificar se o valor 10 é maior que o valor 20.

Como isso não é verdade, então o resultado retornado é o do terceiro argumento.

```
155 -- Exemplo 1: Faça um IF simples para verificar o seu funcionamento.  
156  
157 • SELECT  
158     IF(10 > 20, '10 é maior que 20', '10 não é maior que 20') AS Resposta;  
159  
160  
161  
162
```

	Resposta
▶	10 não é maior que 20

IF (Exemplo 2)

No próximo exemplo, vamos verificar se um determinado funcionário vai receber ou não um bônus pelo seu desempenho.

A regra é simples: se a nota for maior ou igual a 7, o funcionário recebe um bônus de 10% (0.1), caso contrário, não recebe bônus (0).

O resultado final está mostrado ao lado.

```

161 -- Exemplo 2: Uma empresa oferece um bônus de 10% para todos os funcionários que tiveram
162 uma avaliação do RH de acordo com a seguinte regra.
163 -- NotaDesempenho >= 7 -----> Recebe bônus de 10%
164 -- NotaDesempenho < 7 -----> Não recebe bônus
165 • SET @varNotaDesempenho = 8.5;
166
167 • SELECT
168     IF(@varNotaDesempenho >= 7, 0.1, 0) AS Resultado;
169
170

```

	Resultado
▶	0.1

IF (Exemplo 3)

No próximo exemplo, vamos complementar o exemplo anterior e criar uma nova categoria de bônus.

Quem não tiver uma nota acima de 7, ainda poderá receber um bônus se tiver a nota acima de 5.

Perceba que agora, se um funcionário ficar com 6.5, por exemplo, ele não ganha o bônus de 10%, mas ganha o de 5%.

```

161 -- Exemplo 2: Uma empresa oferece um bônus de 10% para todos os funcionários que tiveram
162 uma avaliação do RH de acordo com a seguinte regra.
163 -- NotaDesempenho >= 7 ----> Recebe bônus de 10%
164 -- NotaDesempenho >= 5 ----> Recebe bônus de 5%
165 -- NotaDesempenho < 7 ----> Não recebe bônus
166 • SET @varNotaDesempenho = 6.5;
167
168 • SELECT
169     IF(@varNotaDesempenho >= 7, 0.1, IF(@varNotaDesempenho >= 5, 0.05, 0)) AS Resultado;
170

```

Resultado	
▶	0.05

IFNULL

A função IFNULL tem um funcionamento bem simples.

Ela pede dois argumentos. Se o primeiro argumento informado for NULL, então a função vai retornar o valor no segundo argumento, caso contrário vai retornar o próprio primeiro argumento.

Na imagem ao lado temos um exemplo de aplicação desta função.

Observe que para todos os Telefones que eram diferentes de NULL, a função retornou o próprio telefone da Loja.

Agora para a Loja em que o Telefone era NULL, ela retornou o telefone alternativo especificado no segundo argumento da função

```

172  -- IFNULL
173  -- Exemplo: A tabela de lojas pode ter lojas que não possuem telefone de contato (NULL). Nestes casos, todas
      as lojas que tiverem NULL na coluna Telefone devem ter como ponto de contato o telefone padrão da central de
      atendimento: 0800-999-9999. Utilize a função IFNULL para fazer este ajuste.
174
175 • SELECT
176   *,
177   IFNULL(Telefone, '0800-999-9999') AS 'Telefone Corrigido'
178   FROM lojas;
179

```

ID_Loja	Loja	Gerente	Endereco	Num_Funcionarios	Telefone	Telefone Corrigido
1	Rio de Janeiro	Pedro Martins	R. Visc. de Pirajá, 136 - Ipanema, Rio de Janeiro, RJ, 22451-000	23	(21) 3198-1239	(21) 3198-1239
2	Belo Horizonte	Julia Freitas	Av. Barão Homem de Melo, 1389 - Nova Granada, Belo Horizonte, MG, 31220-000	13	(31) 2891-2394	(31) 2891-2394
3	Salvador	Marcelo Castro	Shopping Barra - Av. Centenário, 2992 - Barra, Salvador, BA, 40310-000	19	(71) 3465-8748	(71) 3465-8748
4	Curitiba	Amanda Lima	R. Brasílio Itiberê, 3279 - Água Verde, Curitiba, PR, 81520-000	31	(41) 2923-2137	(41) 2923-2137
5	Fortaleza	Letícia Gomes	Av. Dom Luís, 500 - Aldeota, Fortaleza - CE, 60450-000	12	(85) 3294-1481	(85) 3294-1481
6	São Paulo	Rogério Lopes	R. Teodoro Sampaio, 954 - Pinheiros, São Paulo, SP, 05403-000	33	(11) 3295-3945	(11) 3295-3945
7	Niterói	Matheus Leal	Rua Quinze de Novembro, 8 - Loja 301a Loja 301a, Niterói, RJ, 24210-000	18	(21) 3134-7263	(21) 3134-7263
8	Goiânia	Carlos Henrique	Av. Goiás Norte, 3.592 - quadra 2.1 - St. Mal. 11, Goiânia, GO, 74010-000	11	NULL	0800-999-9999

ISNULL

A função ISNULL verifica se um determinado valor é NULL. Se for, ela retorna 1 (verdadeiro), caso contrário, retorna 0 (falso).

No exemplo ao lado, utilizamos a função ISNULL para conseguir identificar os clientes que têm o telefone NULL, pois estes vamos precisar verificar o que pode ter acontecido.

```

181  -- ISNULL
182  -- Exemplo: Alguns clientes não cadastraram telefone de contato. Faça uma consulta com uma coluna extra que
     identifique esses clientes de alguma forma.
183
184 • SELECT
185      *,
186      IF(ISNULL(Telefone), 'Verificar', 'Ok') AS Status
187  FROM clientes;
188

```

ID_Cliente	Nome	Sobrenome	Data_Nascimento	Estado_Civil	Sexo	Email	Telefone	Renda_Anual	Qtd_Filhos	Escolaridade	Status
58	Jon	Zhou	1974-03-12	C	M	jon28@hotmail.com	(85) 96675-4865	80000	2	Parcial	Ok
59	Todd	Gao	1974-02-22	C	M	todd14@gmail.com	(11) 96585-5605	80000	2	Parcial	Ok
60	Noah	Powell	1995-08-28	C	M	noah5@yahoo.com.br	(85) 98162-6253	40000	0	Ensino médio	Ok
61	Angela	Murphy	1995-04-02	S	F	angela41@yahoo.com.br	NULL	40000	0	Ensino médio	Verificar
62	Chase	Reed	1995-12-02	C	M	chase21@hotmail.com		40000	0	Ensino médio	Ok
63	Jessica	Henderson	1993-10-04	C	F	jessica29@gmail.com	(21) 93673-6903	60000	0	Parcial	Ok
64	Grace	Butler	1993-11-22	C	F	grace62@gmail.com	NULL	70000	0	Parcial	Verificar
65	Caleb	Carter	1996-09-20	S	M	caleb40@gmail.com	(31) 97809-1800	60000	0	Parcial	Ok
66	Tiffany	Liang	1975-09-18	S	F	tiffany17@gmail.com	(21) 97231-3503	80000	2	Ensino médio	Ok
67	Carolyn	Navarro	1975-09-16	S	F	carolyn30@yahoo.com.br	(21) 94139-6777	80000	2	Ensino médio	Ok
68	Willie	Raii	1975-03-31	C	M	willie40@yahoo.com.br	(71) 97614-1995	80000	2	Ensino médio	Ok

Uma outra função condicional é a NULLIF. Essa função compara duas expressões e retorna NULL se as duas forem iguais. Se forem, retorna NULL; caso contrário, retorna a primeira expressão. Observe na situação 1 que os dois valores de @var1 e @var2 são iguais. Portanto, ao usar a função NULLIF, o resultado é NULL. Já na segunda situação, o valor de @var1 é diferente do valor de @var2. Portanto, nesse caso, o resultado será o valor do primeiro argumento; ou seja, o valor de @var1.

1

```
191 -- NULLIF
192
193 • SET @var1 = 100;
194 • SET @var2 = 100;
195
196 • SELECT
197     NULLIF(@var1, @var2);
198
199
```

	NULLIF(@var1, @var2)
▶	NULL

2

```
191 -- NULLIF
192
193 • SET @var1 = 100;
194 • SET @var2 = 50;
195
196 • SELECT
197     NULLIF(@var1, @var2);
198
199
```

	NULLIF(@var1, @var2)
▶	100

CASE WHEN

O CASE é uma outra forma de tratar condições dentro do MySQL. Abaixo, temos a sua sintaxe:

```
CASE  
    WHEN condicao1 THEN resultado1  
    WHEN condicao2 THEN resultado2  
    WHEN condicao3 THEN resultado3  
    ELSE resultado4  
END;
```

Podemos fazer uma “tradução” da estrutura acima da seguinte forma:

CASO a condição 1 seja verdadeira, **ENTÃO** retorna o resultado1. **CASO** a condição 2 seja verdadeira, **ENTÃO** retorna o resultado2. **CASO** a condição 3 seja verdadeira, **ENTÃO** retorna o resultado3. **CASO CONTRÁRIO**, retorna o resultado4.

CASE WHEN (Exemplo 1)

No primeiro exemplo ao lado, utilizamos a estrutura CASE para substituir, na tabela clientes, as letras M e F pelos textos Masculino e Feminino.

Chamamos de 'Sexo 2' a coluna resultante do CASE.

```

212 -- Exemplo 1: Utilize a estrutura CASE para fazer uma consulta à tabela clientes, substituindo as letras M e F
213 por Masculino e Feminino.
214 • SELECT
215   *,
216   CASE
217     WHEN Sexo = 'M' THEN 'Masculino'
218     ELSE 'Feminino'
219   END AS 'Sexo 2'
220   FROM clientes;
221
222
223

```

ID_Cliente	Nome	Sobrenome	Data_Nascimento	Estado_Civil	Sexo	Email	Telefone	Renda_Anual	Qtd_Filhos	Escolaridade	Sexo 2
1	Ruben	Torres	1985-08-07	C	M	ruben35@hotmail.com	(85) 94132-1176	60000	3	Pós-graduado	Masculino
2	Christy	Zhu	1988-02-10	S	F	christy12@hotmail.com	(41) 96270-6086	70000	0	Pós-graduado	Feminino
3	Elizabeth	Johnson	1988-08-03	S	F	elizabeth5@gmail.com	(31) 92039-5832	80000	5	Pós-graduado	Feminino
4	Julio	Ruiz	1985-07-31	S	M	julio1@hotmail.com	(62) 93391-5891	70000	0	Pós-graduado	Masculino
5	Janet	Alvarez	1985-12-01	S	F	janet9@gmail.com	(21) 93379-3743	70000	0	Pós-graduado	Feminino
6	Marco	Mehta	1984-05-04	C	M	marco14@yahoo.com.br	(85) 99019-3803	60000	3	Pós-graduado	Masculino
7	Rob	Verhoff	1984-07-02	S	F	rob4@hotmail.com	(11) 98162-4760	60000	4	Pós-graduado	Feminino
8	Shannon	Carlson	1984-03-27	S	M	shannon38@gmail.com	(85) 96795-9950	70000	0	Pós-graduado	Masculino

CASE WHEN (Exemplo 2)

No segundo exemplo, queremos categorizar os pedidos de acordo com a sua receita de venda.

Ao lado, temos toda a lógica do exercício e como resolvê-lo.

```

224  -- Exemplo 2: Na tabela pedidos, crie uma estrutura CASE para avaliar as seguintes situações de Receita_Venda:
225  -- Caso Receita_Venda >= 3000 ---> 'Faturamento Alto'
226  -- Caso Receita_Venda >= 1000 ---> 'Faturamento Médio'
227  -- Caso Receita_Venda < 1000 ---> 'Faturamento Baixo'
228
229 • SELECT
230      *,
231      CASE
232          WHEN Receita_Venda >= 3000 THEN 'Faturamento Alto'
233          WHEN Receita_Venda >= 1000 THEN 'Faturamento Médio'
234          ELSE 'Faturamento Baixo'
235      END AS Resultado
236  FROM pedidos;

```

	ID_Pedido	Data_Venda	ID_Loja	ID_Produto	ID_Cliente	Qtd_Vendida	Receita_Venda	Custo_Venda	Custo_Unit	Preco_Unit	Resultado
▶	1	2019-01-01	2	4	16	1	350	129.5	129.5	350	Faturamento Baixo
	2	2019-01-01	1	4	17	1	350	129.5	129.5	350	Faturamento Baixo
	3	2019-01-01	3	4	23	1	350	129.5	129.5	350	Faturamento Baixo
	4	2019-01-01	8	4	24	1	350	129.5	129.5	350	Faturamento Baixo
	5	2019-01-01	4	4	25	1	350	129.5	129.5	350	Faturamento Baixo
	6	2019-01-01	4	4	26	1	350	129.5	129.5	350	Faturamento Baixo
	7	2019-01-01	2	6	14	1	1800	540	540	1800	Faturamento Médio
	8	2019-01-01	8	6	15	1	1800	540	540	1800	Faturamento Médio
	9	2019-01-01	7	6	38	1	1800	540	540	1800	Faturamento Médio

CASE + AND e OR

Podemos combinar a estrutura CASE com o AND ou o OR para criar condições mais complexas.

A sintaxe dessas combinações é mostrada abaixo.

AND

CASE

```
WHEN condicao1 AND condicao2 THEN resultado1  
ELSE resultado2
```

END;

Podemos fazer uma “tradução” da estrutura acima da seguinte forma:

**CASO a condição 1 seja verdadeira E TAMBÉM a condição 2 seja verdadeira,
ENTÃO retorna o resultado1. CASO CONTRÁRIO, retorna o resultado2.**

OR

CASE

```
WHEN condicao1 OR condicao2 THEN resultado1  
ELSE resultado2
```

END;

Podemos fazer uma “tradução” da estrutura acima da seguinte forma:

**CASO a condição 1 seja verdadeira OU a condição 2 seja verdadeira, ENTÃO
retorna o resultado1. CASO CONTRÁRIO, retorna o resultado2.**

CASE + AND (Exemplo)

O enunciado do primeiro exemplo é mostrado abaixo. Precisaremos utilizar a combinação CASE + AND para resolvê-lo.

```
-- APLICAÇÃO AND
-- Exemplo 1. A empresa está com uma ação de dia das mães/dia dos pais. Como vai funcionar?

-- Caso o cliente seja do sexo Feminino E tenha filhos, receberá ofertas de dia das mães
-- Caso o cliente seja do sexo Masculino E tenha filhos, receberá ofertas de dia dos pais
-- Caso contrário, não receberá oferta

-- As ofertas serão enviadas por e-mail, por isso o setor responsável precisa de uma tabela identificando
facilmente quem receberá qual oferta:

-- 'Oferta dia das mães' ---> Mulher com filhos
-- 'Oferta dia dos pais' ---> Homem com filhos
-- 'Sem oferta'           ---> Clientes sem filhos

-- Você deve criar uma coluna extra identificando cada status.
```

CASE + AND (Exemplo)

A estrutura final da solução é mostrada ao lado.

A lógica é basicamente o seguinte:

Quando o sexo do cliente for Feminino E (AND) a quantidade de filhos maior que zero, então a promoção é de dia das mães.

Agora se o sexo do cliente for Masculino E (AND) a quantidade de filhos maior que zero, então a promoção é de dia dos pais.

Caso contrário, não há uma oferta disponível.

```

35 • SELECT
36      *,
37      CASE
38          WHEN Sexo = 'F' AND Qtd_Filhos > 0 THEN 'Oferta dia das mães'
39          WHEN Sexo = 'M' AND Qtd_Filhos > 0 THEN 'Oferta dia dos pais'
40          ELSE 'Sem oferta'
41      END AS 'Situação'
42  FROM clientes;
43
44

```

ID_Cliente	Nome	Sobrenome	Data_Nascimento	Estado_Civl	Sexo	Email	Telefone	Renda_Annual	Qtd_Filhos	Escolaridade	Situação
62	Chase	Reed	1995-12-02	C	M	jessica29@gmail.com	(21) 93673-6903	60000	0	Parcial	Sem oferta
63	Jessica	Henderson	1993-10-04	C	F	jessica29@gmail.com	(21) 93673-6903	60000	0	Parcial	Sem oferta
64	Grace	Butler	1993-11-22	C	F	grace62@gmail.com	(21) 93673-6903	70000	0	Parcial	Sem oferta
65	Caleb	Carter	1996-09-20	S	M	caleb40@gmail.com	(31) 97809-1800	60000	0	Parcial	Sem oferta
66	Tiffany	Liang	1975-09-18	S	F	tiffany17@gmail.com	(21) 97231-3803	80000	2	Ensino médio	Oferta dia das mães
67	Carolyn	Navarro	1975-09-16	S	F	carolyn30@yahoo.com.br	(21) 94139-6777	80000	2	Ensino médio	Oferta dia das mães
68	Willie	Raji	1975-03-31	C	M	wille40@yahoo.com.br	(71) 97614-1995	80000	2	Ensino médio	Oferta dia dos pais
69	Linda	Serrano	1975-06-21	S	F	linda31@hotmail.com	(21) 92428-2938	80000	2	Ensino médio	Oferta dia das mães
70	Casey	Luo	1975-02-01	S	F	casey6@yahoo.com.br	(31) 96609-9916	80000	2	Ensino médio	Oferta dia das mães
71	Amy	Ye	1976-08-09	S	F	amy16@hotmail.com	(41) 92958-3807	70000	2	Ensino médio	Oferta dia das mães
72	Levi	Arun	1976-08-23	S	M	levi6@gmail.com	(71) 92754-9983	70000	2	Ensino médio	Oferta dia dos pais
73	Felicia	Jimenez	1977-11-11	S	F	felicia4@gmail.com	(85) 95647-1806	80000	2	Ensino médio	Oferta dia das mães
74	Blake	Anderson	1977-07-08	S	M	blake9@gmail.com	(11) 98232-2736	80000	2	Ensino médio	Oferta dia dos pais
75	Leah	Ye	1977-09-14	S	F	leah7@hotmail.com	(31) 95663-3082	80000	2	Ensino médio	Oferta dia das mães

CASE + OR (Exemplo)

O enunciado do segundo exemplo é mostrado abaixo. Precisaremos utilizar a combinação CASE + OR para resolvê-lo.

```
-- APLICAÇÃO OR
-- Exemplo 2. A empresa está com uma parceria com as empresas das marcas 'DELL' e 'SAMSUNG'. Isso significa
que os produtos dessas marcas receberão um desconto de 15% em seu custo de aquisição. Faça uma consulta que
retorne uma coluna extra de desconto no custo de aquisição de cada produto.
```

CASE + OR (Exemplo)

Neste novo exemplo, queremos oferecer um desconto de 15% para os produtos de qualquer uma das marcas: DELL ou SAMSUNG.

Para isso, usamos a combinação do CASE + OR.

O resultado final é mostrado ao lado.

	ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit	Desconto?
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966	821,1	
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SN-001939	2800	960	833	
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90	90	
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129,5	110,075	
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109,2	92,82	
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540	540	
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147F1	3100	1395	1395	
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258	258	
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327,6	327,6	
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275	275	
11	Microfone Condensador com Tripé	5	BLUE	MIC-BL-819455	800	344	344	
12	Microfone de mesa com fio condensa...	5	BLUE	MIC-BL-761411	650	214,5	214,5	
13	Notebook LC2100 Intel Core i5 8GB	6	SAMSUNG	NOT-SN-918457	3400	850	722,5	
14	Notebook Inspiron 15 5000 4GB	6	DELL	NOT-DL-000012	3100	1209	1027,649...	

MÓDULO 11

VIEWS

VIEWS

VIEWS



VIEWS: Introdução

Até aqui vimos como criar diferentes consultas aos bancos de dados. Utilizamos comandos como o SELECT, GROUP BY, JOINs, etc para criar tabelas como a mostrada ao lado.

Mas pra onde foram todas essas “tabelas” que a gente criou? Elas estão em algum lugar?

A resposta é: elas não estão em nenhum lugar!

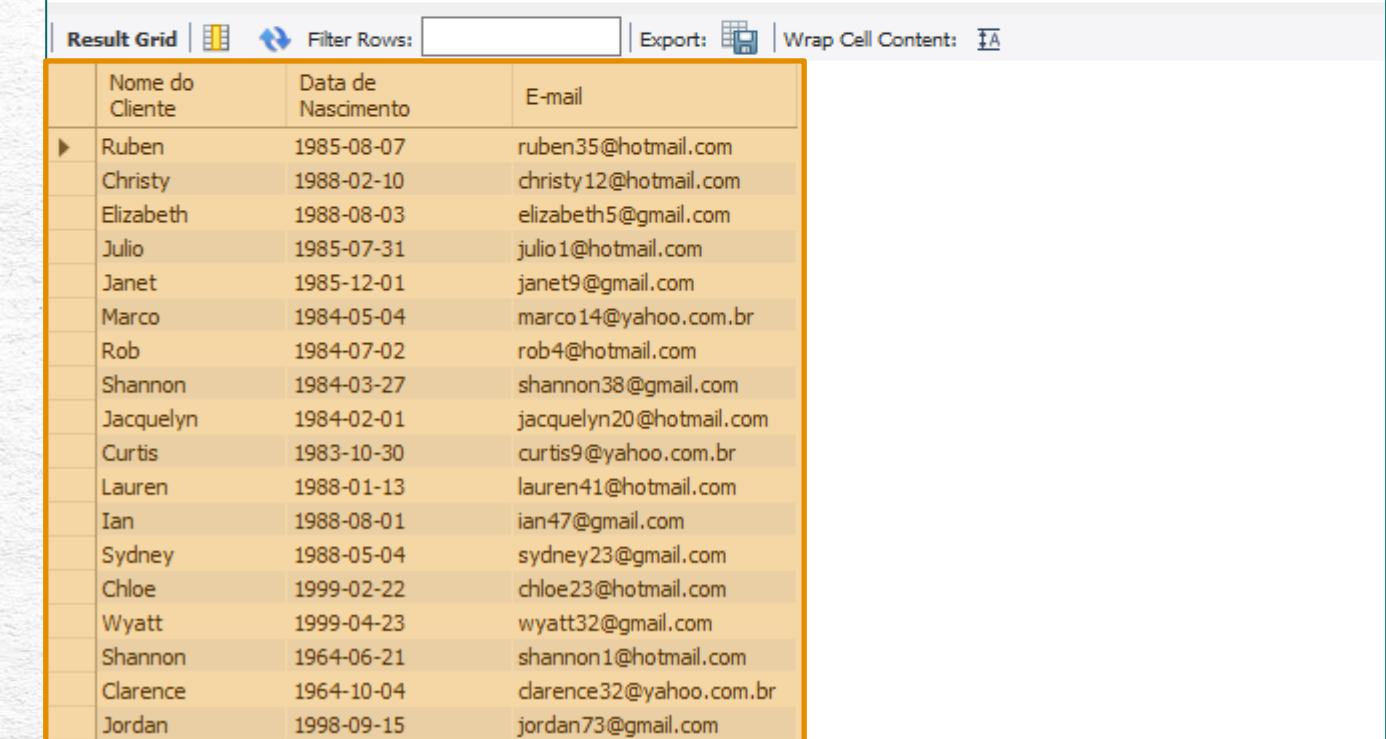
Tudo o que fizemos até agora foi apenas visualizar alguns dados das nossas tabelas do Banco de Dados, nada além disso. Quando executamos um SELECT e logo em seguida executamos outro SELECT, o resultado do primeiro é perdido.

Nenhuma das consultas que fizemos ficou salvo em algum lugar. Inclusive, diversas vezes precisamos criar as mesmas consultas, pois elas se perdem a cada novo SELECT, ou quando fechamos uma consulta e abrimos uma nova.

Existe uma solução pra gente conseguir salvar essas tabelas em algum lugar, e essa solução é a **View**.

```

1 • SELECT
2     Nome AS 'Nome do Cliente',
3     Data_Nascimento AS 'Data de Nascimento',
4     Email AS 'E-mail'
5 FROM clientes;
  
```



	Nome do Cliente	Data de Nascimento	E-mail
▶	Ruben	1985-08-07	ruben35@hotmail.com
	Christy	1988-02-10	christy12@hotmail.com
	Elizabeth	1988-08-03	elizabeth5@gmail.com
	Julio	1985-07-31	julio1@hotmail.com
	Janet	1985-12-01	janet9@gmail.com
	Marco	1984-05-04	marco14@yahoo.com.br
	Rob	1984-07-02	rob4@hotmail.com
	Shannon	1984-03-27	shannon38@gmail.com
	Jacquelyn	1984-02-01	jacquelyn20@hotmail.com
	Curtis	1983-10-30	curtis9@yahoo.com.br
	Lauren	1988-01-13	lauren41@hotmail.com
	Ian	1988-08-01	ian47@gmail.com
	Sydney	1988-05-04	sydney23@gmail.com
	Chloe	1999-02-22	chloe23@hotmail.com
	Wyatt	1999-04-23	wyatt32@gmail.com
	Shannon	1964-06-21	shannon1@hotmail.com
	Clarence	1964-10-04	clarence32@yahoo.com.br
	Jordan	1998-09-15	jordan73@gmail.com

O que é uma View?

- Uma View (ou traduzindo, uma exibição), é uma tabela virtual criada a partir de uma consulta a uma ou mais tabelas (ou até mesmo de outras views) no banco de dados.
- Ela contém linhas e colunas assim como uma tabela real. Nela podemos utilizar comandos como o JOIN, WHERE e outras funções.
- Sempre mostra resultados atualizados dos dados, ou seja, uma vez criada uma View, caso haja alterações no Banco de Dados, as Views são atualizadas automaticamente.
- Caso o servidor seja desligado (ou o SSMS fechado), a view continua armazenada no sistema.

Através de uma View, conseguimos armazenar o resultado de um SELECT (como por exemplo, a tabela à direita) e acessar sempre que precisar, como se fosse uma tabela, com a vantagem de não precisar criar esse SELECT do zero.

1 • SELECT

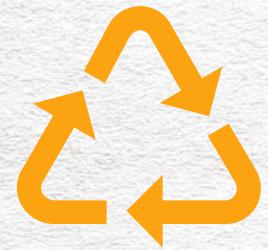
```
1      Nome AS 'Nome do Cliente',
2      Data_Nascimento AS 'Data de Nascimento',
3      Email AS 'E-mail'
4
5  FROM clientes;
```

Result Grid Filter Rows: [] Export: [] Wrap Cell Content: []			
	Nome do Cliente	Data de Nascimento	E-mail
▶	Ruben	1985-08-07	ruben35@hotmail.com
	Christy	1988-02-10	christy12@hotmail.com
	Elizabeth	1988-08-03	elizabeth5@gmail.com
	Julio	1985-07-31	julio1@hotmail.com
	Janet	1985-12-01	janet9@gmail.com
	Marco	1984-05-04	marco14@yahoo.com.br
	Rob	1984-07-02	rob4@hotmail.com
	Shannon	1984-03-27	shannon38@gmail.com
	Jacquelyn	1984-02-01	jacquelyn20@hotmail.com
	Curtis	1983-10-30	curtis9@yahoo.com.br
	Lauren	1988-01-13	lauren41@hotmail.com
	Ian	1988-08-01	ian47@gmail.com
	Sydney	1988-05-04	sydney23@gmail.com
	Chloe	1999-02-22	chloe23@hotmail.com
	Wyatt	1999-04-23	wyatt32@gmail.com
	Shannon	1964-06-21	shannon1@hotmail.com
	Clarence	1964-10-04	clarence32@yahoo.com.br
	Jordan	1998-09-15	jordan73@gmail.com

Por que criar uma View?

São muitas as vantagens de uma View. Abaixo temos algumas das principais:

Reutilização



Sempre que necessário, podemos consultar aquela View, pois ela fica armazenada no sistema.

Segurança



Ao criar uma View, estamos ocultando linhas ou colunas da tabela original do banco de dados. Desta forma, apenas algumas informações relevantes serão visualizadas na View.

Ganho de tempo



Quando criamos Views, estamos poupando o tempo de recriar vários SELECTs, o que aumenta a produtividade.

CREATE VIEW: Criando a primeira view

Para criar uma View, utilizamos o comando **CREATE VIEW**. Na imagem abaixo temos a estrutura padrão:

```
CREATE VIEW nome_da_view AS
SELECT
    Coluna1,
    Coluna2,
    Coluna3
FROM
    Tabela
```

```
1 • CREATE VIEW vwClientes AS
2   SELECT
3     Nome AS 'Nome do Cliente',
4     Data_Nascimento AS 'Data de Nascimento',
5     Email AS 'E-mail'
6   FROM clientes;
```

Uma vez criada, a View ficará armazenada no banco de dados, na pasta de Views.

E para selecionar essa View, utilizamos o comando **SELECT**.

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database schema for the 'base' schema. It shows tables like 'categorias', 'clientes', 'locais', 'lojas', 'pedidos', 'produtos', and views like 'vwclientes'. An orange arrow points from the 'vwclientes' view in the Navigator to the 'Result Grid' on the right. The 'Result Grid' contains the following data:

Nome do Cliente	Data de Nascimento	E-mail
Ruben	1985-08-07	ruben35@hotmail.com
Christy	1988-02-10	christy12@hotmail.com
Elizabeth	1988-08-03	elizabeth5@gmail.com
Julio	1985-07-31	julio1@hotmail.com
Janet	1985-12-01	janet9@gmail.com
Marco	1984-05-04	marco14@yahoo.com.br
Rob	1984-07-02	rob4@hotmail.com
Shannon	1984-03-27	shannon38@gmail.com
Jacquelyn	1984-02-01	jacquelyn20@hotmail.com
Curtis	1983-10-30	curtis9@yahoo.com.br
Lauren	1988-01-13	lauren41@hotmail.com

ALTER VIEW: Alterando a view criada

Para alterar uma View criada, usamos o comando **ALTER VIEW**.

Na imagem ao lado, temos um exemplo. A **vwClientes**, criada anteriormente, foi alterada para considerar apenas os clientes do sexo feminino.

```

1 • ALTER VIEW vwClientes AS
2   SELECT
3     Nome AS 'Nome do Cliente',
4     Data_Nascimento AS 'Data de Nascimento',
5     Email AS 'E-mail',
6     Sexo AS 'Sexo'
7   FROM clientes
8   WHERE Sexo = 'F';
9
10 • SELECT * FROM vwclientes;
  
```

Result Grid Filter Rows: [] Export: [] Wrap Cell Content: []				
	Nome do Cliente	Data de Nascimento	E-mail	Sexo
▶	Christy	1988-02-10	christy12@hotmail.com	F
	Elizabeth	1988-08-03	elizabeth5@gmail.com	F
	Janet	1985-12-01	janet9@gmail.com	F
	Rob	1984-07-02	rob4@hotmail.com	F
	Jacquelyn	1984-02-01	jacquelyn20@hotmail.com	F
	Lauren	1988-01-13	lauren41@hotmail.com	F
	Sydney	1988-05-04	sydney23@gmail.com	F
	Chloe	1999-02-22	chloe23@hotmail.com	F
	Shannon	1964-06-21	shannon1@hotmail.com	F
	Destiny	1998-08-29	destiny7@hotmail.com	F
	Jill	1966-04-06	jill13@gmail.com	F
	Bethany	1967-02-17	bethany10@hotmail.com	F
	Theresa	1967-03-17	theresa13@hotmail.com	F

ALTER VIEW: Alterando a view criada

Para alterar a view também podemos clicar nela com o botão direito e ir na opção Alter View. A janela que se abre mostra inclusive o código que deu origem à view criada.

The screenshot illustrates the process of modifying a view in MySQL Workbench. On the left, the 'SCHEMAS' tree shows the 'base' schema with various tables and views. The 'vwclientes' view is selected and has a context menu open. The 'Alter View...' option is highlighted. On the right, the 'Query 1' window displays the SQL code for the 'vwclientes' view:

```

Query 1  vwclientes - View x
Name: vwclientes
The name of the view is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:
1 • CREATE
2     ALGORITHM = UNDEFINED
3     DEFINER = `root`@`localhost`
4     SQL SECURITY DEFINER
5     VIEW `vwclientes` AS
6     SELECT
7         `clientes`.`Nome` AS `Nome do Cliente`,
8         `clientes`.`Data_Nascimento` AS `Data de Nascimento`,
9         `clientes`.`Email` AS `E-mail`,
10        `clientes`.`Sexo` AS `Sexo`
11    FROM
12        `clientes`
13    WHERE
14        (`clientes`.`Sexo` = 'F')

```

DROP VIEW: Excluindo uma view

Para excluir uma View criada, usamos o comando **DROP VIEW**.

```
1  DROP VIEW vwclientes;
```

O comando é bem simples e é ilustrado na imagem ao lado.

Criando uma View com o comando WHERE (Exemplo 1)

No exemplo a seguir vamos criar uma View que combina o comando SELECT com o WHERE.

```
1 # Exemplos Views
2
3 # 1. VIEWS + WHERE: Criando Views com consultas filtradas.
4
5 -- Exemplo 1. Crie uma View chamada vwReceitaAcima4000 que armazene todas as colunas da tabela
   Pedidos. A sua View deverá conter apenas as vendas com receita acima de R$4.000.
```

Criando uma View com o comando WHERE (Exemplo 1)

Aplicando o que aprendemos até então, o resultado final do código é mostrado abaixo.

```
5 -- Exemplo 1. Crie uma View chamada vwReceitaAcima4000 que armazene todas as colunas da tabela
6   Pedidos. A sua View deverá conter apenas as vendas com receita acima de R$4.000.
7
8 • SELECT * FROM pedidos;
9
10 • CREATE VIEW vwReceitaAcima4000 AS
11   SELECT
12     *
13   FROM pedidos
14   WHERE Receita_Venda >= 4000;
```

Criando uma View com o comando WHERE (Exemplo 2)

No exemplo 2 vamos mais uma vez criar uma View que utilize o comando WHERE. O enunciado do exemplo é mostrado abaixo.

```
-- Exemplo 2. Crie uma View chamada vwProdutosAtualizada que armazene todas as colunas da tabela Produtos. A sua view deverá conter apenas os produtos das marcas DELL, SAMSUNG e SONY.
```

Criando uma View com o comando WHERE (Exemplo 2)

Aplicando o que aprendemos até então, o resultado final do código é mostrado abaixo.

```
-- Exemplo 2. Crie uma View chamada vwProdutosAtualizada que armazene todas as colunas da tabela Produtos. A sua view deverá conter apenas os produtos das marcas DELL, SAMSUNG e SONY.  
• CREATE VIEW vwProdutosAtualizada AS  
SELECT  
*  
FROM produtos  
WHERE Marca_Produto IN ('DELL', 'SAMSUNG', 'SONY');
```

Criando uma View com o comando GROUP BY

Agora vamos ver um exemplo de como criar uma View utilizando o GROUP BY.

1. VIEWS + GROUP BY: Criando Views com consultas filtradas.

-- Exemplo 1. Crie uma view que será o resultado de um agrupamento da tabela de pedidos. A ideia é que você tenha nessa view o total de Receita e Custo agrupado por ID_Produto.

Criando uma View com o comando GROUP BY

Aplicando o que já sabemos, o gabarito é mostrado abaixo. Observe que não há nenhuma grande diferença na criação das views com comandos como o WHERE, GROUP BY, etc. Sempre começaremos com o CREATE VIEW, e em seguida o código da consulta que queremos realizar.

1. VIEWS + GROUP BY: Criando Views com consultas filtradas.

-- Exemplo 1. Crie uma view que será o resultado de um agrupamento da tabela de pedidos. A ideia é que você tenha nessa view o total de Receita e Custo agrupado por ID_Produto.

```
CREATE VIEW vwReceitaECustoTotal AS
SELECT
    ID_Produto,
    SUM(Receita_Venda) AS 'Total Receita',
    SUM(Custo_Venda) AS 'Total Custo'
FROM pedidos
GROUP BY ID_Produto;
```

Alterando a View criada com o WHERE

Criada uma View, podemos alterá-la usando o comando ALTER VIEW. Abaixo, temos um exemplo de como incluir o comando WHERE para trabalharmos em conjunto com o GROUP BY.

```
-- Exemplo 2. Altere a view anterior para mostrar o agrupamento apenas para os produtos da loja 2.

ALTER VIEW vwReceitaECustoTotal AS
SELECT [REDACTED]
    ID_Produto,
    SUM(Receita_Venda) AS 'Total Receita',
    SUM(Custo_Venda) AS 'Total Custo'
FROM pedidos [REDACTED]
WHERE ID_Loja = 2
GROUP BY ID_Produto;
```

Alterando a View criada com o HAVING

Agora, podemos alterar nossa view usando o comando ALTER VIEW para realizar um filtro com o HAVING, conforme mostrado no exemplo abaixo.

```
-- Exemplo 3. Altere a view anterior para mostrar o agrupamento apenas para os produtos que tiveram uma  
receita total maior que 1 milhão, na loja 2.  
  
ALTER VIEW vwReceitaECustoTotal AS  
SELECT  
    ID_Produto,  
    SUM(Receita_Venda) AS 'Total Receita',  
    SUM(Custo_Venda) AS 'Total Custo'  
FROM pedidos  
WHERE ID_Loja = 2  
GROUP BY ID_Produto  
HAVING SUM(Receita_Venda) >= 1000000;
```

Criando uma View com o comando JOIN

Vejamos agora um exemplo de View onde usamos o comando Join.

```
# 1. VIEWS + JOIN e GROUP BY: Aplicando Join nas views criadas.
```

```
-- Exemplo 1. Crie uma view que seja a junção entre as tabelas de pedidos e de produtos. Ou seja, essa view deve conter todas as colunas da tabela pedidos e as colunas Nome_Produto, Marca_Produto e Num_Serie da tabela de produtos.
```

Criando uma View com o comando JOIN

Mais uma vez, nenhuma grande novidade na criação da View. Podemos aplicar exatamente o que já sabemos sobre Joins e chegar no resultado abaixo.

1. VIEWS + JOIN e GROUP BY: Aplicando Join nas views criadas.

-- Exemplo 1. Crie uma view que seja a junção entre as tabelas de pedidos e de produtos. Ou seja, essa view deve conter todas as colunas da tabela pedidos e as colunas Nome_Produto, Marca_Produto e Num_Serie da tabela de produtos.

```
CREATE VIEW vwPedidosCompleta AS
SELECT
    pe.*,
    pr.Nome_Produto,
    pr.Marca_Produto,
    pr.Num_Serie
FROM pedidos AS pe
INNER JOIN produtos AS pr
    ON pe.ID_Produto = pr.ID_Produto;
```

Criando uma View com o comando JOIN + GROUP BY

Por fim, temos um exemplo de criação de View utilizando o comando Join combinado com o Group By.

```
-- Exemplo 2. Crie uma view que será o resultado de um agrupamento da tabela de pedidos. A ideia é que você  
tenha nessa view o total de Receita e Custo agrupados por Nome_Produto.  
  
• CREATE VIEW vwResultadoFinal AS  
  SELECT  
    pr.Nome_Produto,  
    SUM(Receita_Venda) AS 'Receita_Total',  
    SUM(Custo_Venda) AS 'Custo_Total'  
  FROM pedidos AS pe  
  INNER JOIN produtos AS pr  
  ON pe.ID_Produto = pr.ID_Produto  
  GROUP BY Nome_Produto;
```

MÓDULO 12

CRUD

CRUD

CRUD



O que significa CRUD?

Operações CRUD são operações que conseguimos fazer em um Banco de Dados. Essa sigla significa o seguinte:



CREATE

Permite criar Bancos de Dados, Tabelas ou Exibições (Views)



READ

Permite ler os dados do banco de dados, tabelas e views.



UPDATE

Permite atualizar os dados do banco de dados, tabelas ou views.



DELETE

Permite excluir dados de um banco de dados, tabelas e views.

No módulo anterior, vimos como criar Views (exibições) com os operadores CRUD. Essas exibições não são entendidas exatamente como tabelas do banco de dados. Agora o que vamos fazer é aprender como criar tabelas propriamente ditas.

CRUD em bancos de dados

Podemos utilizar comandos SQL para criar, mostrar, selecionar e excluir bancos de dados.

CREATE DATABASE

SHOW DATABASES

USE

SELECT DATABASE

DROP DATABASE

Permite criar bancos de dados

Mostra os bancos de dados

Define um banco de dados como o padrão

Seleciona o banco de dados padrão

Exclui um banco de dados

A seguir, vejamos como utilizar esses comandos na prática.

CREATE DATABASE

Para criar um banco de dados, usamos o comando `CREATE DATABASE`, como mostrado em 1.

Podemos complementar esse comando usando o `IF NOT EXISTS` para que o banco de dados seja criado apenas se ele já não existir, como mostrado em 2. A vantagem é que, se o banco de dados já existir, não será retornado um erro.

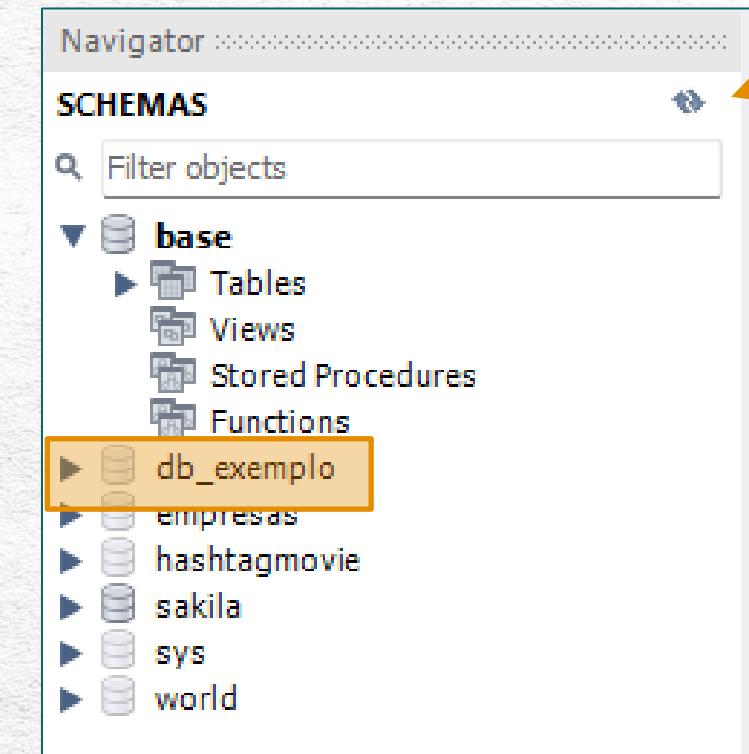
Por fim, em 3, clique no botão de atualizar indicado pela seta para visualizar o banco de dados criado.

`CREATE DATABASE db_Exemplo;`

1

`CREATE DATABASE IF NOT EXISTS db_Exemplo;`

2



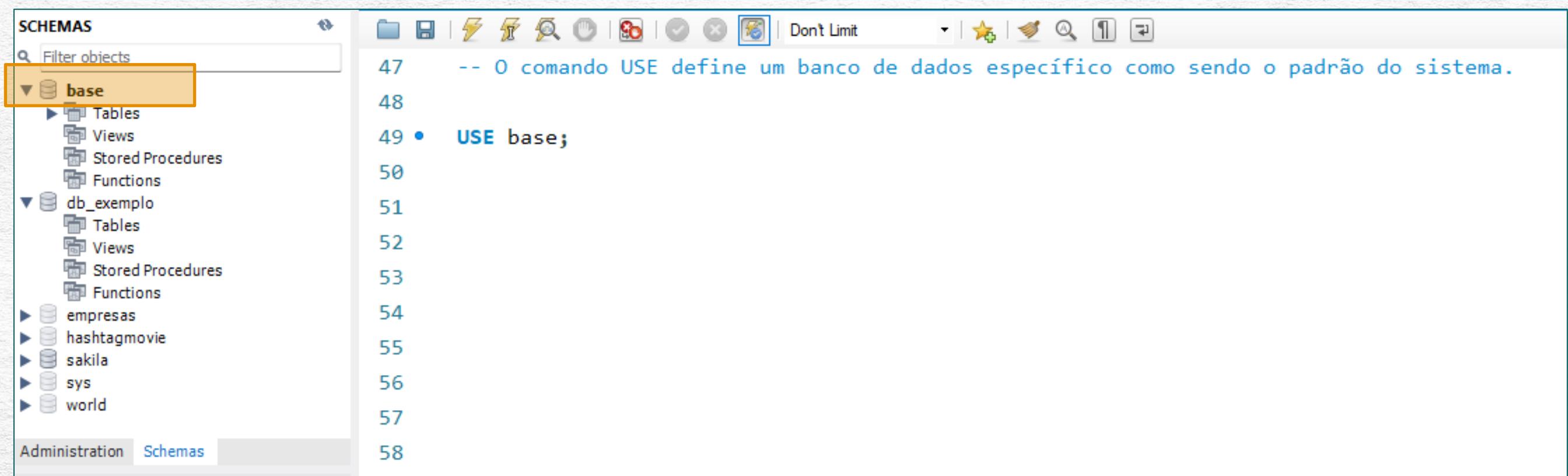
3

SHOW DATABASES

Com o comando SHOW DATABASES podemos listar todos os bancos de dados existentes.

43 • SHOW DATABASES;	
	Database
▶	base
	db_exemplo
	empresas
	hashtagmovie
	information_schema
	mysql
	performance_schema
	sakila
	sys
	world

Utilizamos o comando USE para definir como padrão um determinado banco de dados. Basta usar o comando USE seguido do nome do banco de dados, conforme exemplo abaixo.



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' browser displays a list of databases: 'base' (selected and highlighted with an orange border), 'db_exemplo', 'empresas', 'hashtagmovie', 'sakila', 'sys', and 'world'. The 'base' database is expanded, showing its tables, views, stored procedures, and functions. On the right, the SQL editor contains the following code:

```
47 -- O comando USE define um banco de dados específico como sendo o padrão do sistema.  
48  
49 • USE base;  
50  
51  
52  
53  
54  
55  
56  
57  
58
```

DROP DATABASE

Para excluir um banco de dados, utilizamos o comando **DROP DATABASE**, seguido do nome do banco de dados que queremos excluir, conforme exemplo abaixo.

```
74      -- Para excluir um bando de dados, usamos o seguinte comando:  
75  
76 •  DROP DATABASE IF EXISTS db_Exemplo;  
77
```

CRUD em tabelas

Podemos utilizar comandos SQL para criar, mostrar, selecionar, atualizar, incluir dados e excluir tabelas.

CREATE TABLE

SHOW TABLES

SELECT

INSERT INTO

UPDATE

Permite criar uma tabela

Mostra as tabelas do banco de dados

Permite selecionar tabelas

Insere dados dentro de tabelas

Atualiza dados em uma tabela

TRUNCATE

DROP

Exclui todas as linhas de uma tabela

Exclui uma tabela

A seguir, vejamos como utilizar esses comandos na prática.

CREATE DATABASE

Primeiramente, vamos recriar o banco de dados db_Exemplo para que possamos adicionar dentro dele as novas tabelas. Utilize o código abaixo.

Lembre-se do comando USE para tornar esse banco de dados o padrão do sistema.

```
-- Crie o banco de dados db_Exemplo e defina-o como padrão:  
CREATE DATABASE IF NOT EXISTS db_Exemplo;  
USE db_Exemplo;  
--
```

Quando criamos uma nova tabela, precisamos especificar quais são as colunas que essa tabela deve conter. Cada uma dessas colunas vai armazenar um tipo de dados específico. Os principais tipos de dados são listados abaixo:

TIPO	DESCRIÇÃO
INT	Inteiros entre -2.147.483.648 e 2.147.483.647
DECIMAL(M, D)	Número decimal com M dígitos no total e D casas decimais. O padrão é (10, 2). Exemplo: o valor 100,34 poderia ser declarado como DECIMAL(5, 2).
FLOAT(M, D)	Número decimal com M dígitos no total e D casas decimais. O padrão é (10, 2). Exemplo: o valor 12345,67 poderia ser declarado como FLOAT(7, 2).
VARCHAR(N)	String de tamanho variável, até 65535 caracteres.
DATE	Uma data de 01/01/1000 a 31/12/9999, no formato YYYY-MM-DD
DATETIME	Uma combinação de data e hora de 01/01/1000 00:00:00 a 31/12/9999 23:59:59, no formato YYYY-MM-DD HH:MM:SS

CREATE TABLE

Agora vamos criar as tabelas do nosso banco de dados.

O banco será composto por 3 tabelas, que vão armazenar informações de uma empresa que oferece treinamentos online. As tabelas criadas armazenarão os seguintes dados:

- 1) Alunos
- 2) Cursos
- 3) Matrículas

Ao lado, temos a estrutura para criação de cada uma das tabelas.

```

• CREATE TABLE IF NOT EXISTS dAlunos(
    ID_Aluno INT,
    Nome_Aluno VARCHAR(100),
    Email VARCHAR(100)
);

• CREATE TABLE IF NOT EXISTS dCursos(
    ID_Curso INT,
    Nome_Curso VARCHAR(100),
    Preco_Curso DECIMAL(10, 2)
);

• CREATE TABLE IF NOT EXISTS fMatriculas(
    ID_Matricula INT,
    ID_Aluno INT,
    ID_Curso INT,
    Data_Cadastro DATE
);

```

CREATE TABLE

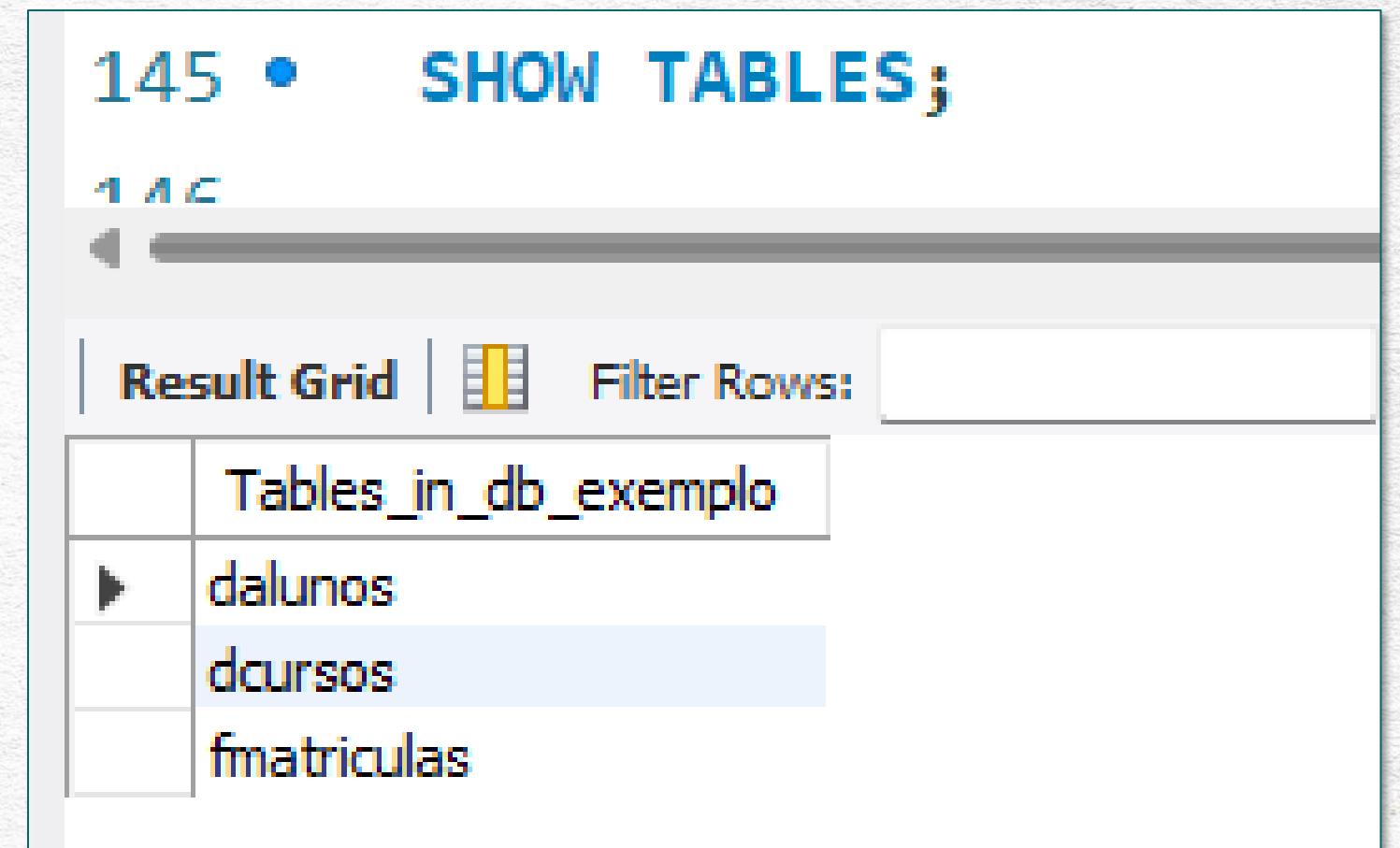
A estrutura será sempre:

- 1) O comando CREATE TABLE
- 2) Comando IF NOT EXISTS (opcional)
- 3) Nome da tabela
- 4) Nome da coluna seguido do tipo daquela coluna

```
• ⏪ CREATE TABLE IF NOT EXISTS dAlunos(  
    ID_Aluno INT,  
    Nome_Aluno VARCHAR(100),  
    Email VARCHAR(100)  
);
```

SHOW TABLES

Para mostrar as tabelas de um banco de dados, usamos o comando SHOW TABLES, conforme mostrado ao lado.



The screenshot shows a MySQL Workbench interface. At the top, there is a status bar with the number '145' and a message 'SHOW TABLES;'. Below the status bar is a toolbar with icons for 'Result Grid' (selected), 'Filter Rows', and other options. The main area is a 'Result Grid' table with one column labeled 'Tables_in_db_exemplo'. The table contains four rows with the names of the tables: 'dalunos', 'dcursos', and 'fmatriculas'. The row for 'dcursos' is currently selected, indicated by a blue highlight.

Tables_in_db_exemplo
dalunos
dcursos
fmatriculas

CONSTRAINTS

Quando criamos tabelas, podemos definir regras para os valores adicionados naquelas tabelas.

É aqui que entram as Restrições (Constraints). Elas se tratam de regras aplicadas nas colunas de uma tabela, proporcionando um melhor controle sobre os dados adicionados naquelas tabelas.

Exemplo 1: Podemos especificar que uma coluna não pode ter valores NULL.

Exemplo 2: Podemos especificar que uma coluna deverá ser uma chave primária ou chave estrangeira.

Basicamente, as constraints são usadas para limitar os tipos de dados que são inseridos e temos alguns tipos de constraints que veremos a seguir.

Abaixo, temos listadas as principais constraints usadas na criação de tabelas.

NOT NULL

UNIQUE

DEFAULT

PRIMARY KEY

FOREIGN KEY

CONSTRAINTS

Abaixo, temos listadas as principais constraints usadas na criação de tabelas.

NOT NULL

UNIQUE

DEFAULT

PRIMARY KEY

FOREIGN KEY

A constraint NOT NULL faz com que uma coluna não aceite valores NULL. Um valor NULL é diferente de zero ou vazio, ele identifica que nenhum valor foi definido.

A constraint NOT NULL obriga um campo a sempre possuir um valor. Dessa forma, não é possível inserir um registro ou atualizar sem entrar com um valor neste campo.

Nome_Cliente VARCHAR(100) NOT NULL

CONSTRAINTS

Abaixo, temos listadas as principais constraints usadas na criação de tabelas.

NOT NULL

UNIQUE

DEFAULT

PRIMARY KEY

FOREIGN KEY

A restrição UNIQUE impede que em uma mesma coluna sejam adicionados valores repetidos.

Podemos usar esse tipo de constraint em uma coluna de CPF, por exemplo. Dois clientes não podem ter um mesmo CPF, por isso podemos usar a constraint UNIQUE na coluna CPF para impedir que valores duplicados sejam cadastrados.

CPF VARCHAR(100) UNIQUE

CONSTRAINTS

Abaixo, temos listadas as principais constraints usadas na criação de tabelas.

NOT NULL

UNIQUE

DEFAULT

PRIMARY KEY

FOREIGN KEY

Essa restrição permite definir um valor padrão para uma determinada coluna. Dessa forma, caso um valor não seja adicionado na coluna, automaticamente ela receberá o valor padrão informado.

Cidade VARCHAR(100) DEFAULT 'São Paulo'

CONSTRAINTS

Abaixo, temos listadas as principais constraints usadas na criação de tabelas.

NOT NULL

UNIQUE

DEFAULT

PRIMARY KEY

FOREIGN KEY

A constraint PRIMARY KEY identifica de forma única cada registro em uma tabela do banco de dados.

Chaves primárias devem conter valores únicos. Uma coluna de chave primária não pode conter valores NULL. Cada tabela deve conter 1 e apenas 1 chave primária.

PRIMARY KEY(ID_Curso)

CONSTRAINTS

Abaixo, temos listadas as principais constraints usadas na criação de tabelas.

NOT NULL

UNIQUE

DEFAULT

PRIMARY KEY

FOREIGN KEY

Uma FOREIGN KEY em uma tabela é um campo que aponta para uma chave primária em outra tabela.

FOREIGN KEY(ID_Curso) REFERENCES dCursos(ID_Curso)

Preparação para criar as constraints

Agora que entendemos o que são as constraints, a ideia é recriar as tabelas dAlunos, dCursos e fMatriculas incluindo constraints.

Portanto, o primeiro passo é excluir as tabelas para poder recriá-las. Vamos utilizar o comando DROP TABLE, como mostrado ao lado, para excluir cada uma dessas tabelas.

```
DROP TABLE IF EXISTS dAlunos;  
DROP TABLE IF EXISTS dCursos;  
DROP TABLE IF EXISTS fMatriculas;
```

Recriando a tabela dAlunos

Vamos começar recriando a tabela dAlunos.

Essa tabela terá duas restrições principais:

- 1) A coluna ID_Aluno será uma chave primária (Primary Key).
- 2) As colunas Nome_Aluno e Email não aceitarão valores nulos.

O código final é mostrado ao lado.

```
• CREATE TABLE dAlunos(  
    ID_Aluno INT,  
    Nome_Aluno VARCHAR(100) NOT NULL,  
    Email VARCHAR(100) NOT NULL,  
    PRIMARY KEY(ID_Aluno)  
);
```

Recriando a tabela dCursos

Vamos agora recriar a tabela dCursos.

Essa tabela terá duas restrições principais:

- 1) A coluna ID_Curso será uma chave primária (Primary Key).
- 2) As colunas Nome_Curso e Preco_Curso não aceitarão valores nulos.

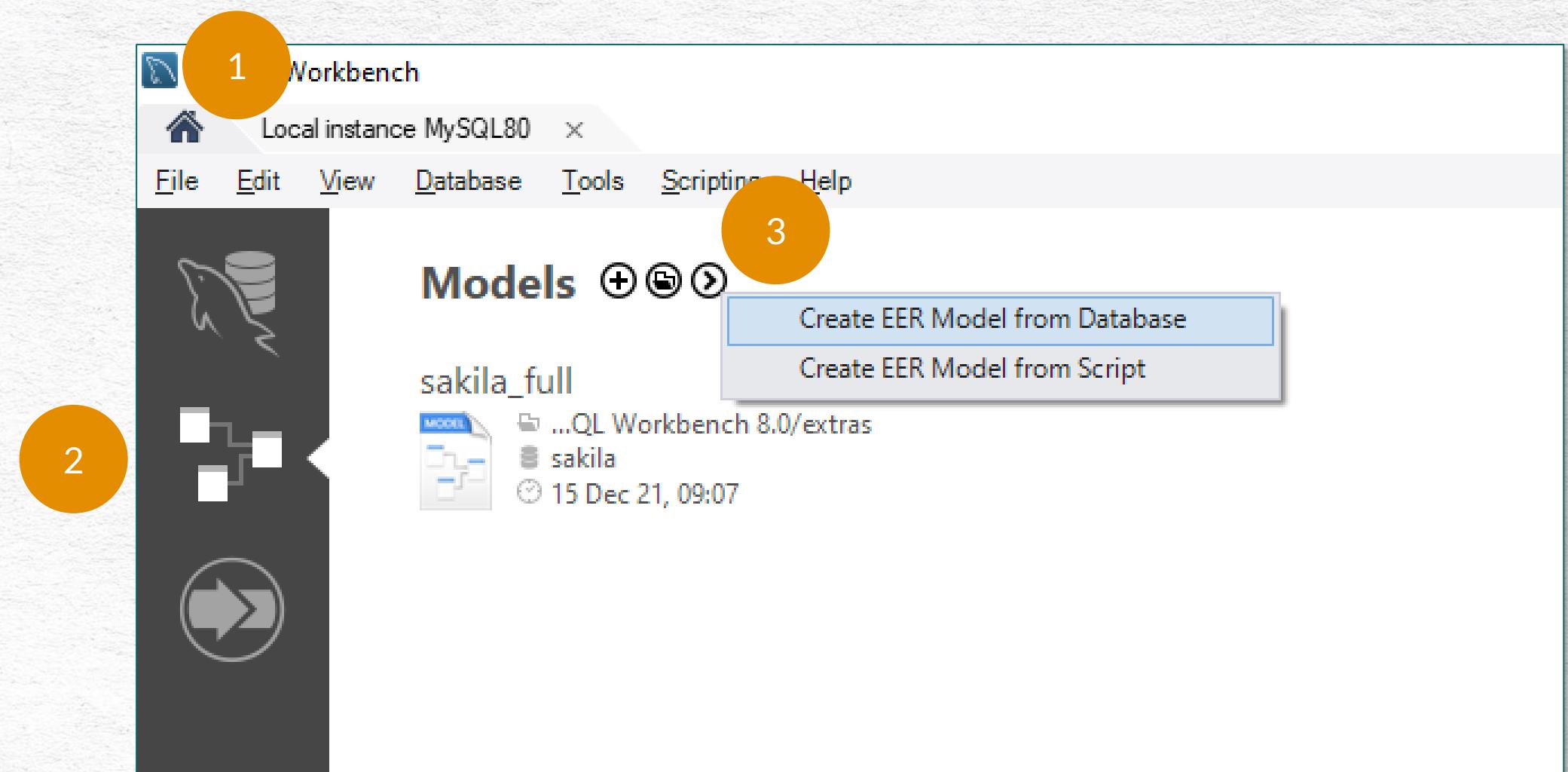
O código final é mostrado ao lado.

```
• CREATE TABLE IF NOT EXISTS dCursos(  
    ID_Curso INT,  
    Nome_Curso VARCHAR(100) NOT NULL,  
    Preco_Curso DECIMAL(10, 2) NOT NULL,  
    PRIMARY KEY(ID_Curso)  
);
```

Visualizando os relacionamentos entre tabelas

Agora vamos ver a diferença prática que a criação de constraints proporciona.

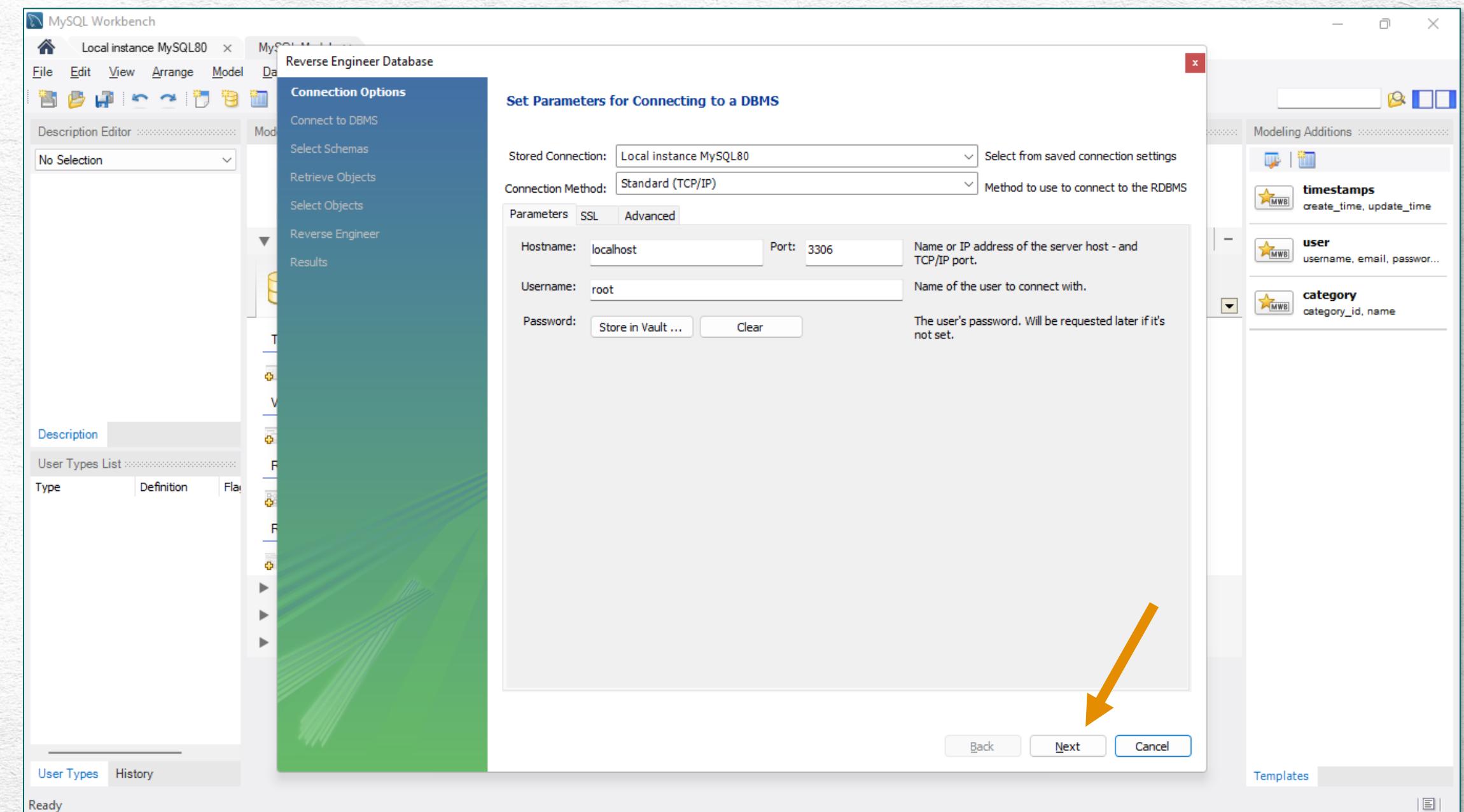
As constraints de primary key e foreign key criadas vão gerar relacionamentos entre essas tabelas, que podemos visualizar através do caminho ao lado.



Clique na casinha (1), depois clique em (2) e por fim, na setinha em (3) clique em 'Create EER Model from Database'.

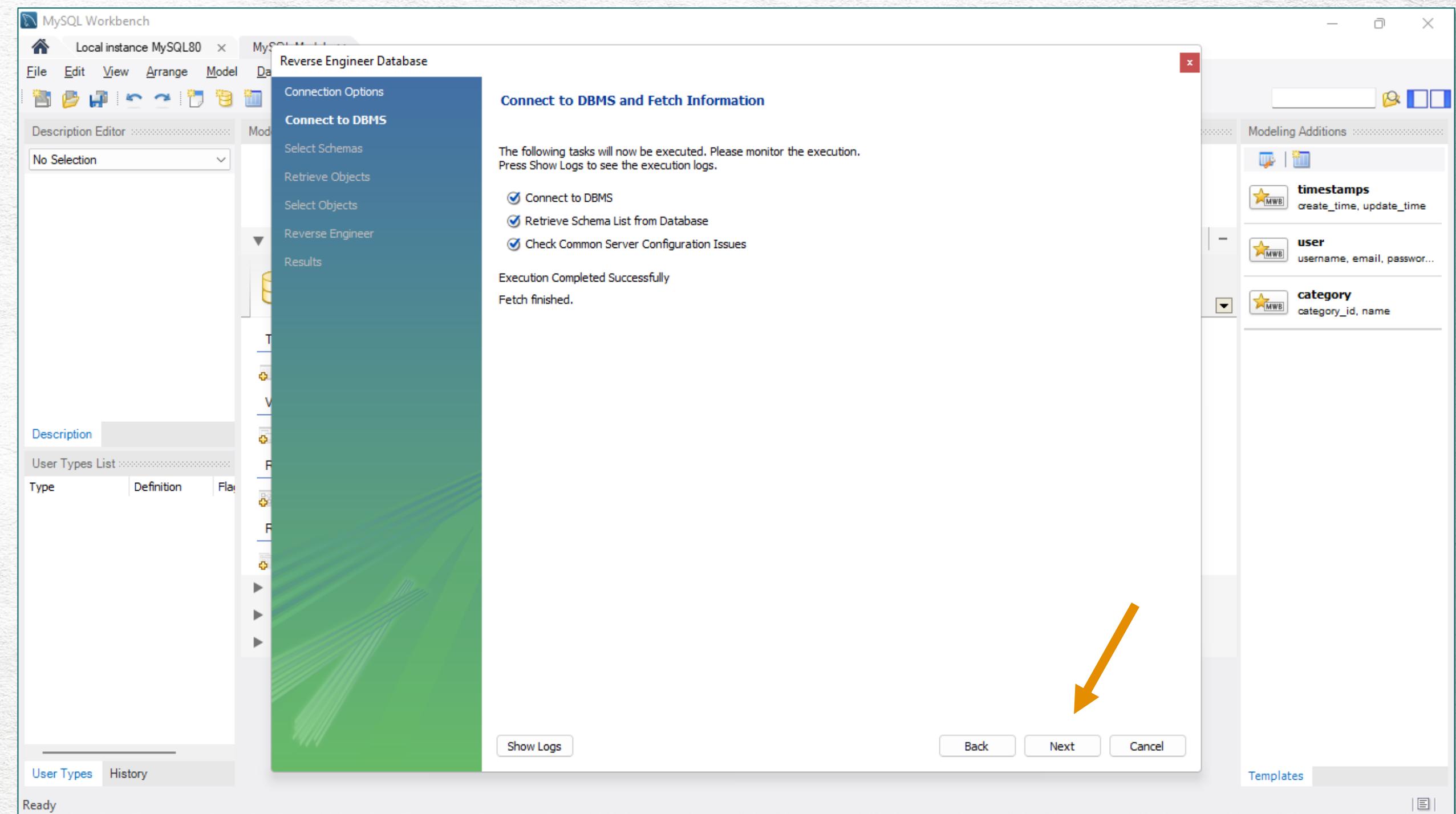
Visualizando os relacionamentos entre tabelas

Em seguida, na janela que abrir,
clique em Next.



Visualizando os relacionamentos entre tabelas

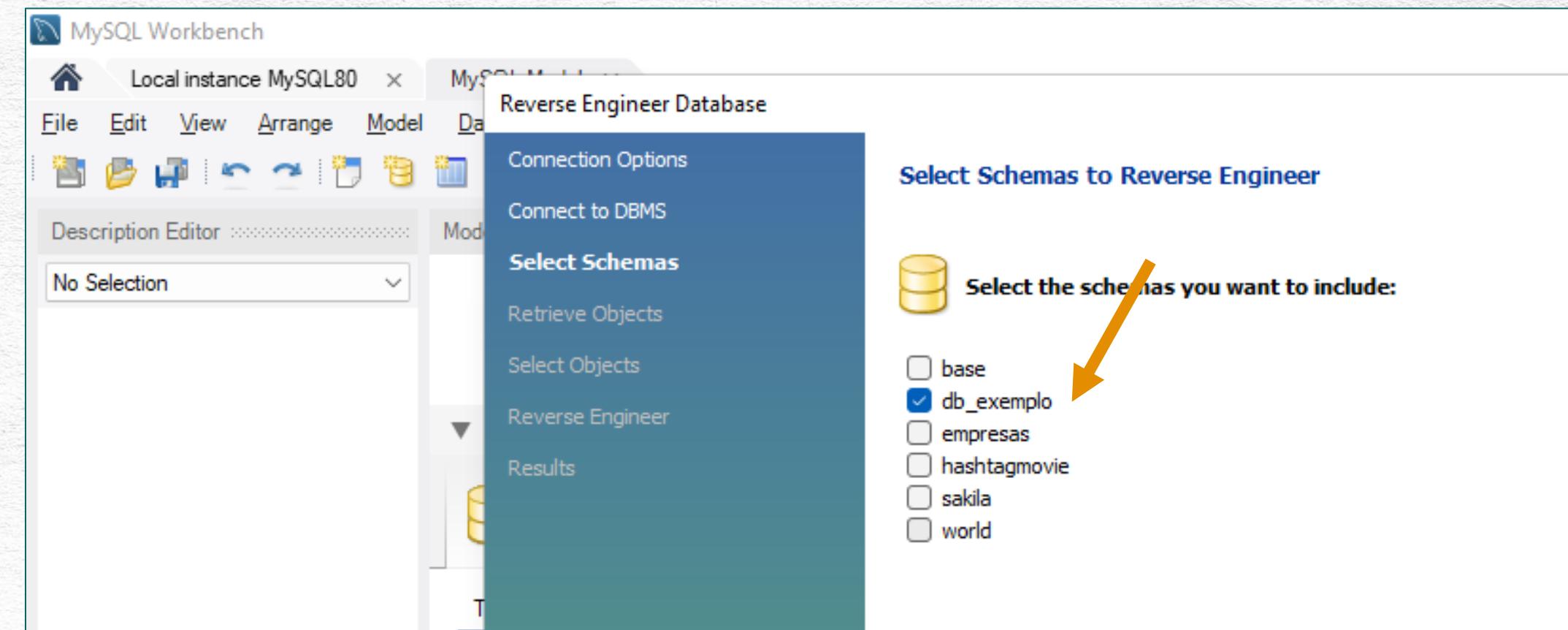
Em seguida, na janela que abrir, clique em Next novamente.



Visualizando os relacionamentos entre tabelas

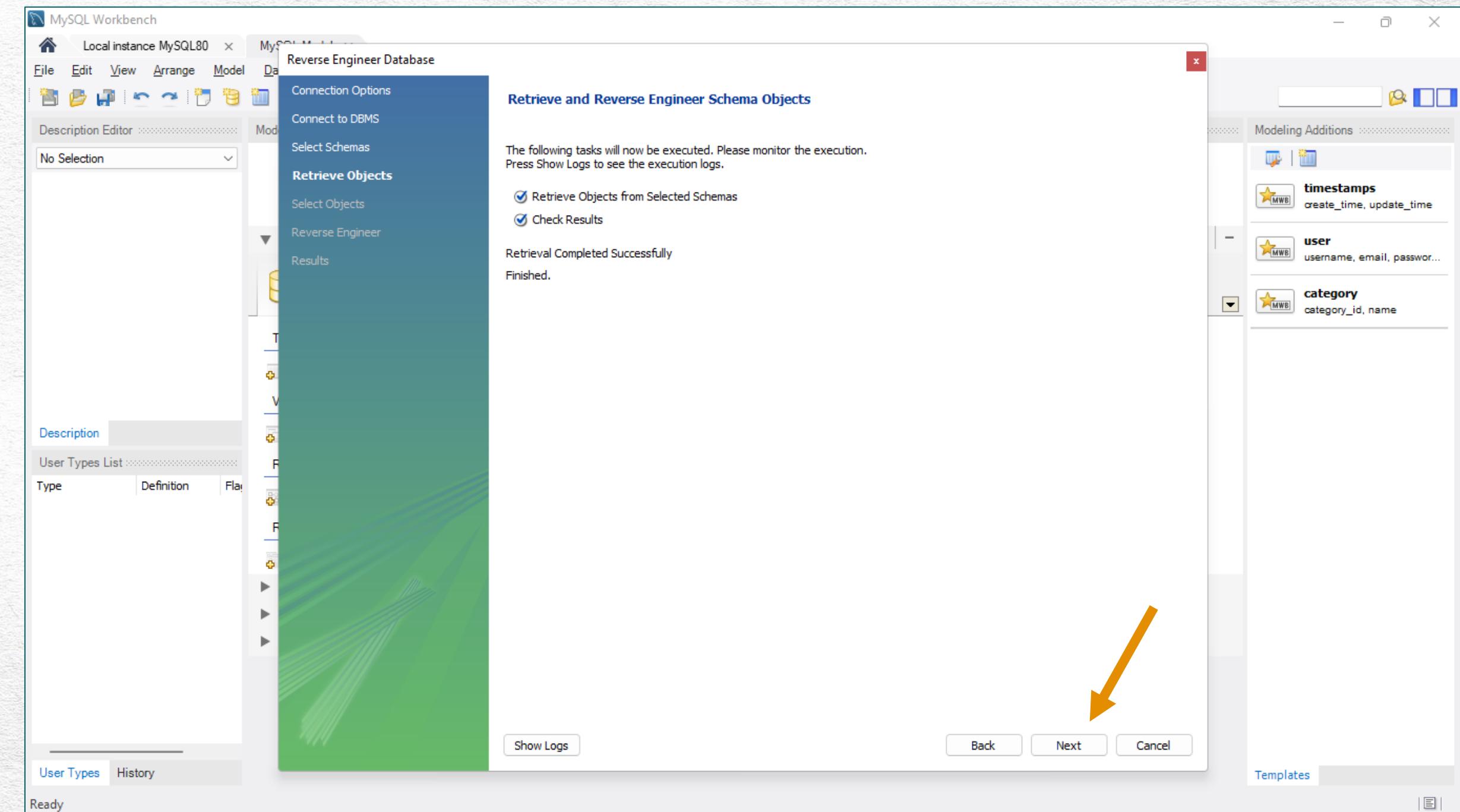
Na lista de bancos de dados, escolha aquele que você deseja visualizar os relacionamentos. No caso, o db_Exemplo.

Depois é só clicar em Next.



Visualizando os relacionamentos entre tabelas

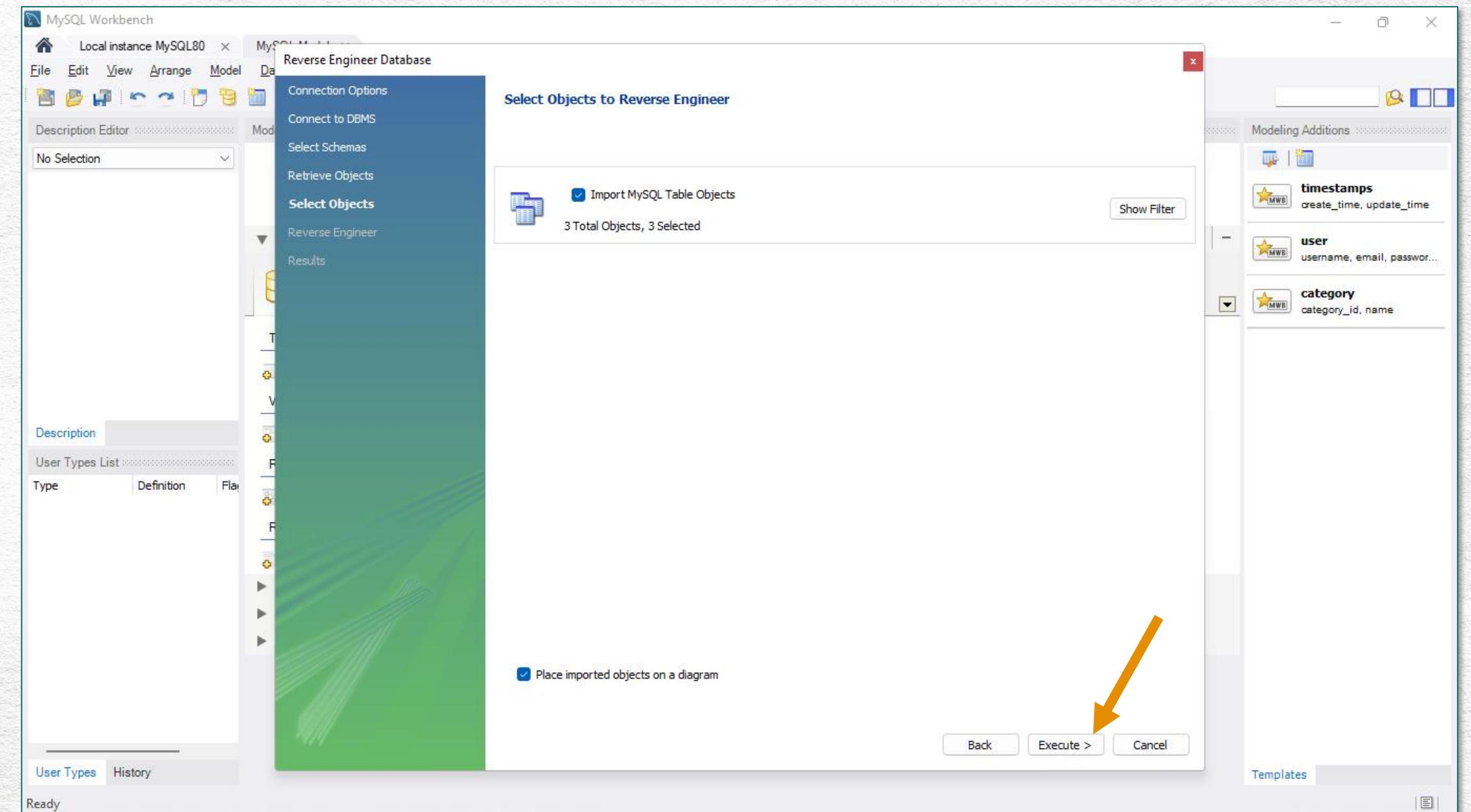
Na janela seguinte, clique em Next novamente.



Visualizando os relacionamentos entre tabelas

Por fim, em Execute.

Depois, é só seguir com o Next até que finalize as etapas.

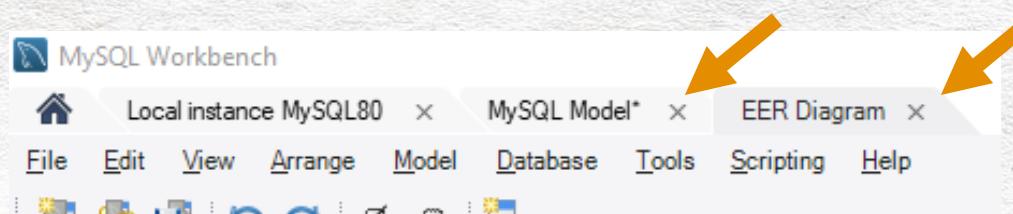
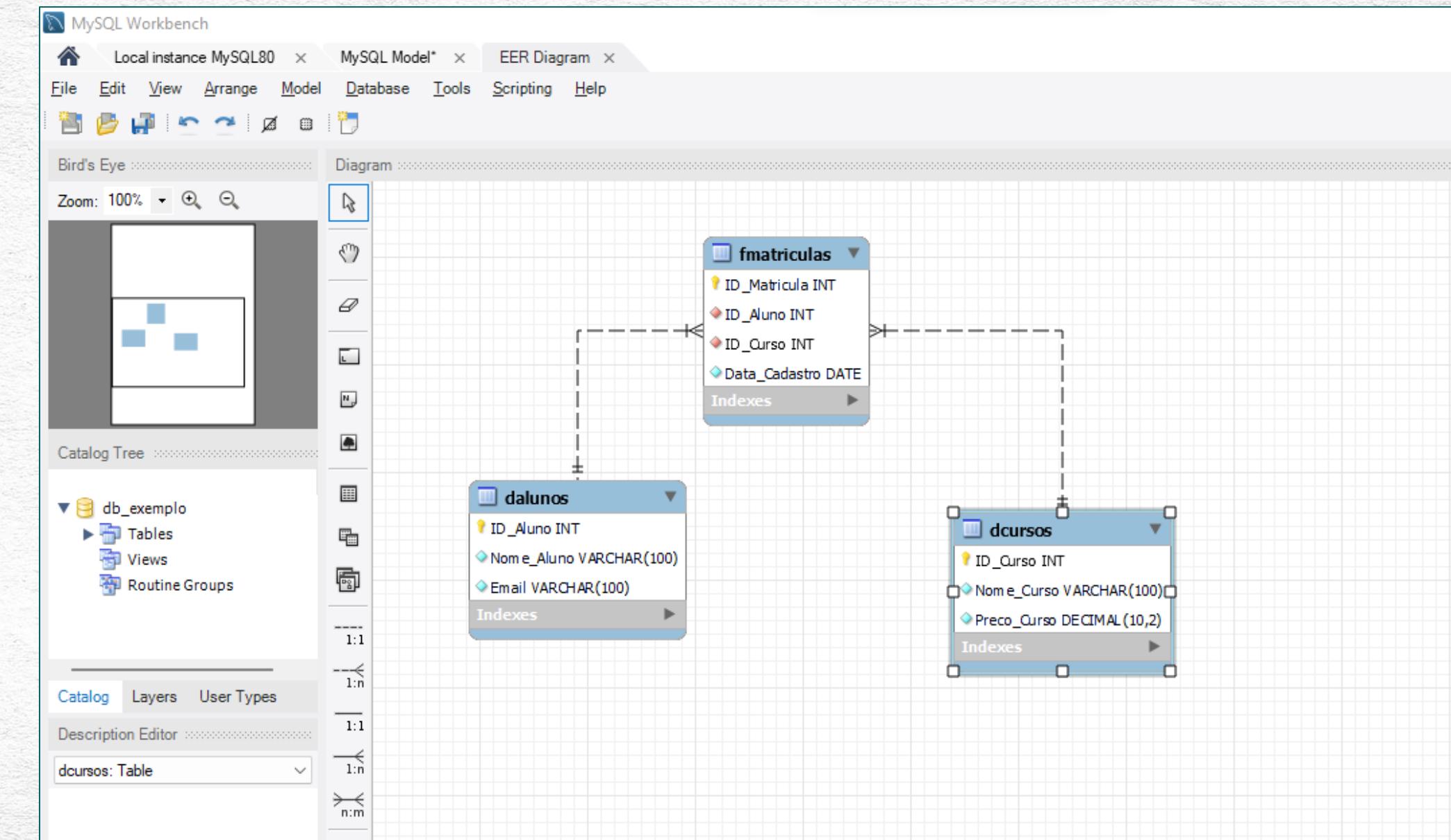


Visualizando os relacionamentos entre tabelas

Ao terminar as etapas, será aberta uma janela com o diagrama onde você poderá visualizar as tabelas do banco, relacionadas entre si.

Essa relação surgiu por conta das restrições de chave primária e chave estrangeira criadas.

Por fim, você pode simplesmente fechar a janela para voltar para o MySQL, onde estamos criando nossos códigos.



INSERT INTO

Agora que criamos as tabelas, vamos adicionar valores nelas usando o comando **INSERT INTO**.

Para cada uma das tabelas, o código é mostrado ao lado.

```

247 # 5. Inserindo dados nas tabelas.
248
249 • INSERT INTO dAlunos(ID_Aluno, Nome_Aluno, Email)
250   VALUES
251     (1, 'Ana' , 'ana123@gmail.com' ),
252     (2, 'Bruno' , 'bruno_vargas@outlook.com' ),
253     (3, 'Carla' , 'carlinha1999@gmail.com' ),
254     (4, 'Diego' , 'diicastroneves@gmail.com' );
255
256 • INSERT INTO dCursos(ID_Curso, Nome_Curso, Preco_Curso)
257   VALUES
258     (1, 'Excel' , 100),
259     (2, 'VBA' , 200),
260     (3, 'Power BI' , 150);
261
262 • INSERT INTO fMatriculas(ID_Matricula, ID_Aluno, ID_Curso, Data_Cadastro)
263   VALUES
264     (1, 1, 1, '2021/03/11' ),
265     (2, 1, 2, '2021/06/21' ),
266     (3, 2, 3, '2021/01/08' ),
267     (4, 3, 1, '2021/04/03' ),
268     (5, 4, 1, '2021/05/10' ),
269     (6, 4, 3, '2021/05/10' );
270

```

UPDATE

Podemos atualizar dados de uma tabela usando o comando UPDATE, conforme a estrutura mostrada ao lado.

Um detalhe importante é que você não pode esquecer de incluir o comando WHERE para garantir que você vai atualizar apenas a linha desejada.

Caso você esqueça do WHERE, todas as linhas da tabela serão atualizadas.

6. Atualizando dados de uma tabela com o UPDATE

- **UPDATE** dCursos
SET Preco_Curso = 300
WHERE ID_Curso = 1;

DELETE

Para excluir linhas da tabela, usamos o DELETE conforme a estrutura ao lado.

Assim como no UPDATE, é muito importante que você especifique a linha que deseja excluir através do WHERE para não correr o risco de excluir todas as linhas da tabela.

```
# DELETE. Deletando registros de uma tabela  
• DELETE FROM fMatriculas  
WHERE ID_Matricula = 6;
```

DROP TABLE x TRUNCATE TABLE

Por fim, temos outros dois comandos que nos permitem excluir dados.

O DROP TABLE é um comando que já vimos e que permite excluir definitivamente uma tabela.

Já o TRUNCATE TABLE é um comando que exclui de uma vez todas as linhas de uma tabela, mas mantém essa tabela no banco de dados.

TRUNCATE TABLE x DROP TABLE

-- TRUNCATE TABLE: Deleta todos os registros da tabela de uma vez, mas a tabela continua existindo.

-- DROP TABLE: Deleta todos os registros da tabela, inclusive a própria tabela.

- TRUNCATE TABLE fMatriculas;
- DROP TABLE fMatriculas;

MÓDULO 13

FUNCTIONS E PROCEDURES

FUNCTIONS E PROCEDURES

FUNCTIONS E PROCEDURES



Functions e Procedures

Se tratam de dois tipos de rotinas. Uma rotina é um conjunto de instruções que você pode salvar no seu banco de dados e executar quando quiser, sem a necessidade de criar o código do zero toda vez que você precisa dele.

Funções (UDFs) e Procedimentos (Stored Procedures) são um pouco similares, mas com algumas diferenças em termos de aplicações.

BEGIN... END;

Os comandos BEGIN e END têm como objetivo delimitar um bloco de comandos a serem executados por uma Function ou Stored Procedure.

Cada bloco BEGIN / END possui um delimitador (;).

Porém, o delimitador ; pode ser problemático pois, ao ser encontrado em um procedimento ou função, ele a finaliza imediatamente. Para não termos problemas, devemos então mudar esse delimitador e para isso usamos o comando DELIMITER para criar rotinas com declarações compostas.

BEGIN... END: Function

Ao lado, temos um exemplo de Function criada, contendo o bloco BEGIN..END.

```
1 -- Criar função para cálculo de receita de produto
2
3 DELIMITER //
4 • CREATE FUNCTION fn_CalculaReceita(Quantidade INT, Preco DECIMAL(10, 2))
5   RETURNS DECIMAL(10, 2) DETERMINISTIC
6   BEGIN
7     RETURN quantidade * preco;
8   END //
9   DELIMITER ;
10
11 • SELECT fn_CalculaReceita(10, 99.90);
12
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	fn_CalculaReceita(10, 99.90)			
▶	999.00			
<				

BEGIN... END: Stored Procedure

Ao lado, temos um exemplo de Procedure criada, contendo o bloco BEGIN..END.

```
14    -- Criar procedimento para busca de preço do produto
15    DELIMITER $$
16 •  CREATE PROCEDURE sp_BuscaPreco(ID INT)
17    BEGIN
18        SELECT CONCAT('O preço do produto é: ', Preco_Unit) AS 'Resultado'
19        FROM produtos
20        WHERE ID_Produto = ID;
21        SELECT 'Procedimento finalizado com sucesso!!' AS 'Mensagem';
22    END $$
23    DELIMITER ;
24
25 •  CALL sp_BuscaPreco(4);
26
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Resultado				
► O preço do produto é: 350				

Escopo das variáveis: Revisão

O escopo de uma variável se refere ao local onde essa variável existe, ou seja, o local onde ela pode ser acessada.

Temos os seguintes níveis de escopo:

1. USER-DEFINED VARIABLES
2. LOCAL VARIABLES

Escopo das variáveis: Revisão

O escopo de uma variável se refere ao local onde essa variável existe, ou seja, o local onde ela pode ser acessada.

Temos os seguintes níveis de escopo:

1. USER-DEFINED VARIABLES
2. LOCAL VARIABLES



USADA DENTRO DE FUNCTIONS E STORED PROCEDURES

Declaração de variáveis locais

Podemos criar variáveis locais em um procedimento ou função usando uma declaração DECLARE dentro de um bloco BEGIN.

A variável pode ser criada e inicializada com um valor se desejado.

Ficam disponíveis apenas dentro do bloco onde foram criadas, e em blocos que existam dentro do bloco onde a variável foi criada.

Após o bloco ter sido executado e encerrado, a variável é desalocada da memória.

FUNCTION

```
DELIMITER $$  
CREATE FUNCTION fn_Exemplo(valor INT)  
RETURNS INT  
BEGIN  
    DECLARE var1 INT;  
    DECLARE var2 DECIMAL(10, 2);  
    DECLARE var3 VARCHAR(100);  
    DECLARE var4 DATE;  
  
    SET var1 = 10;  
    SET var2 = 9.98;  
    SET var3 = 'SQL';  
    SET var4 = '2999-12-31';  
END $$
```

PROCEDURE

```
DELIMITER $$  
CREATE PROCEDURE sp_Exemplo()  
BEGIN  
    DECLARE var1 INT;  
    DECLARE var2 DECIMAL(10, 2);  
    DECLARE var3 VARCHAR(100);  
    DECLARE var4 DATE;  
  
    SET var1 = 10;  
    SET var2 = 9.98;  
    SET var3 = 'SQL';  
    SET var4 = '2999-12-31';  
END $$
```

Functions

Uma função é uma rotina, um conjunto de instruções que você pode salvar no seu banco de dados e executar quando quiser, sem a necessidade de criar o código do zero toda vez que você precisa dele.

A estrutura para criação de uma Function é mostrada abaixo:

DELIMITER \$\$

CREATE FUNCTION nome_function(parametro1 TIPO1, parametro2 TIPO2)

RETURNS TIPO DETERMINISTIC

BEGIN

.... Bloco de códigos

END \$\$

DELIMITER ;

Functions: Exemplo 1

Vamos começar criando a primeira Function, usando o código ao lado.

Essa function será bem simples e retornará uma mensagem de boas vindas.

Podemos chamar uma function diretamente dentro de um SELECT, como mostrado ao lado. Basta informar o parâmetro de entrada para que ela possa retornar o resultado.

```
-- Exemplo 1. Crie uma função que retorna o seguinte texto: "Olá ___, tudo bem?".

DELIMITER $$

CREATE FUNCTION fn_BoasVindas(nome VARCHAR(100))
RETURNS VARCHAR(100) DETERMINISTIC
BEGIN
    RETURN CONCAT('Olá ', nome, ', tudo bem?');
END $$

DELIMITER ;
```

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a query editor window displays the SQL code for creating a function named fn_BoasVindas. The code defines the function to take a parameter nome (VARCHAR(100)) and return a VARCHAR(100) value, being deterministic. It uses the CONCAT function to build a greeting string. After the code, a semicolon is followed by another DELIMITER command. In the main pane below, a single row of results is shown for the query:

1	SELECT fn_BoasVindas('Marcus');
	fn_BoasVindas('Marcus')
	Olá Marcus, tudo bem?

Functions: Exemplo 2

Uma função é usada para gerar um valor que pode ser usado em uma expressão.

O valor (resultado) é gerado baseado em um ou mais parâmetros de entrada fornecidos à função.

```
1 • DROP FUNCTION IF EXISTS fn_Faturamento;  
2  
3 • CREATE FUNCTION fn_Faturamento(preco DECIMAL(10, 2), quantidade INT)  
4   RETURNS DECIMAL(10, 2) DETERMINISTIC  
5   RETURN preco * quantidade;  
6  
7  
8 • SELECT fn_Faturamento(2.5, 10) AS 'Faturamento';
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Faturamento				
▶ 25.00				

Functions: Exemplo 3

Uma função é usada para gerar um valor que pode ser usado em uma expressão.

O valor (resultado) é gerado baseado em um ou mais parâmetros de entrada fornecidos à função.

```

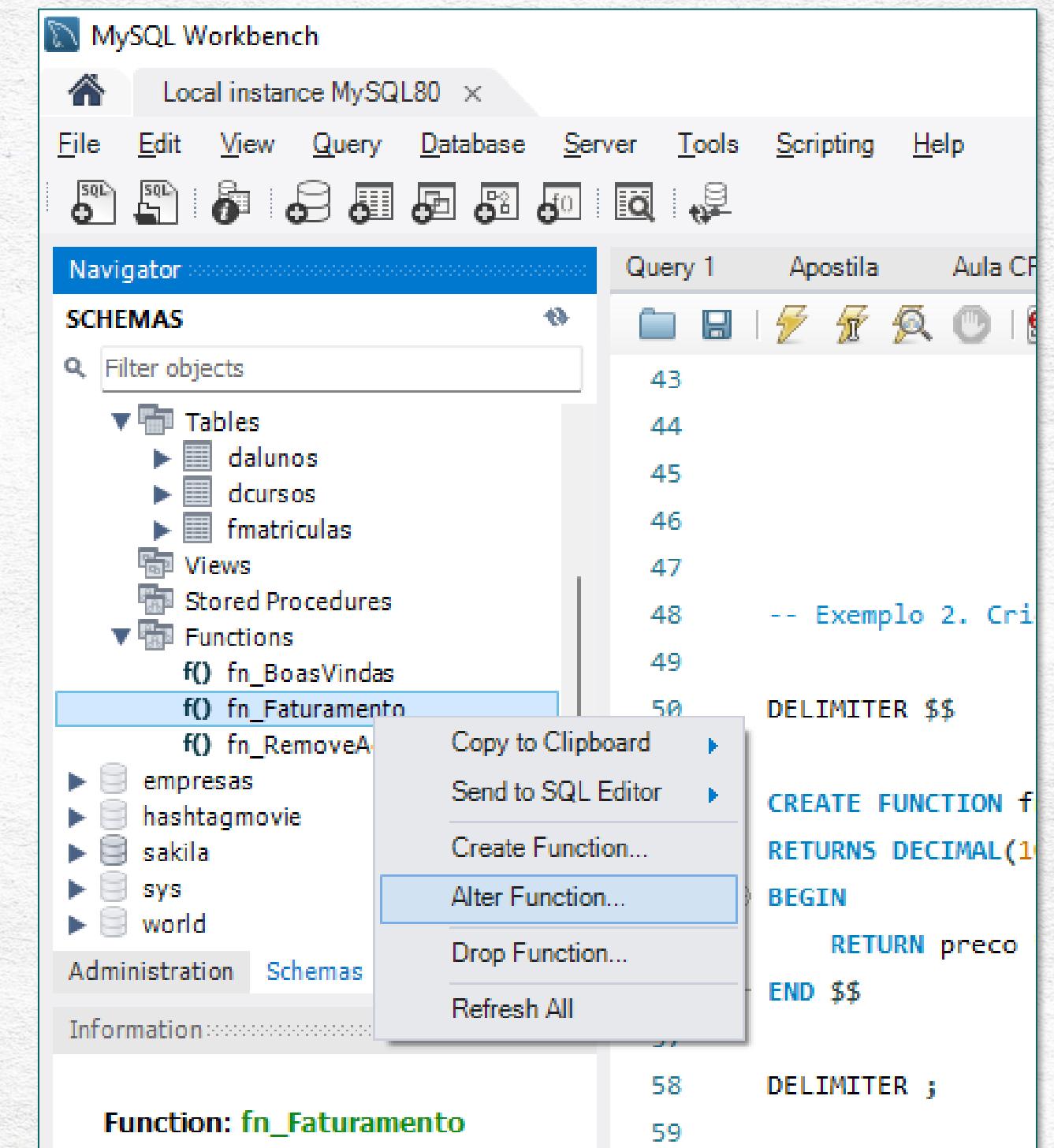
1 • CREATE FUNCTION fn_CalculaDesconto(preco DECIMAL(10, 2), desconto DECIMAL(10, 2))
2   RETURNS DECIMAL(10, 2) DETERMINISTIC
3   RETURN preco * (1 - desconto);
4
5 • SELECT Nome_Produto, Preco_Unit, fn_CalculaDesconto(Preco_Unit, 0.5) FROM produtos;

```

	Nome_Produto	Preco_Unit	fn_CalculaDesconto(Preco_Unit, 0.5)
▶	Monitor LED 19,5" Full HD HDMI	2300	1150.00
	Monitor Curvo 24" 144Hz HDMI	2800	1400.00
	Webcam Full HD 1080p	450	225.00
	Kit Teclado + Mouse sem fio Wireless	350	175.00
	Kit Teclado + Mouse Slim Bluetooth	280	140.00
	Cadeira Gamer reclinável Azul/Laranja	1800	900.00

Alterando uma function

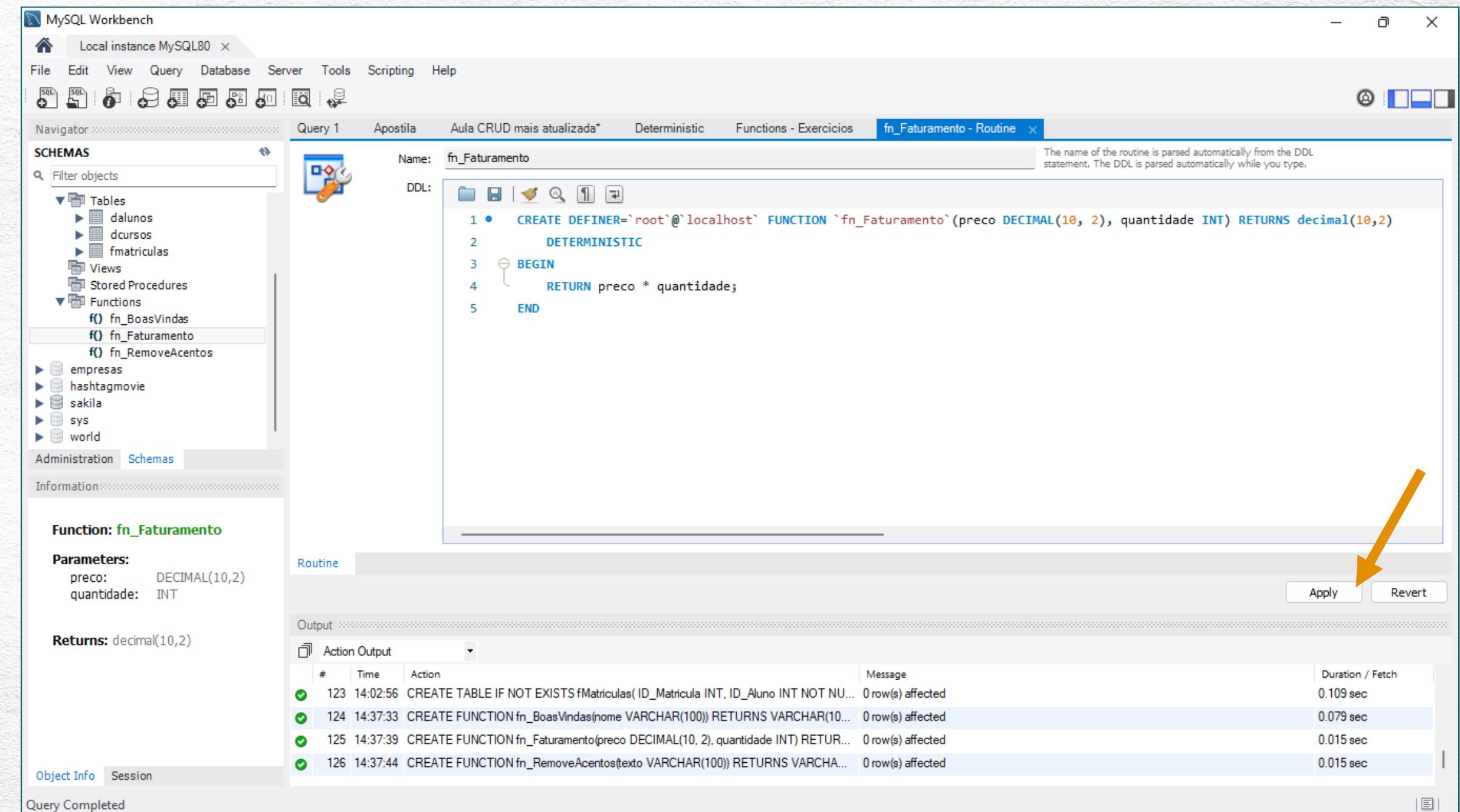
Para alterar uma Function, do lado esquerdo, na lista de Functions criadas, clique com o botão direito na que você deseja alterar e no menu que aparecer, utilize a opção Alter Function.



Alterando uma function

Feito isso, uma janela se abrirá onde você pode visualizar a Function criada, e consequentemente alterar essa function.

Após as alterações, basta clicar em Apply (conforme indicado pela seta ao lado).



Para que serve o DETERMINISTIC

Ao criar uma função armazenada, você deve declarar que ela é determinística ou que não modifica os dados do banco de dados. Caso contrário, pode ser inseguro para recuperação ou replicação de dados.

Caso a gente não declare a function como DETERMINISTIC, será retornado o erro 1418.

Para corrigir este problema, temos dois caminhos:

- 1) Seguir utilizando o DETERMINISTIC como fizemos até agora.
- 2) Utilizar o código abaixo para dizer para o MySQL que as funções criadas são confiáveis.

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

Stored Procedures: (Procedimentos Armazenados)

As Procedures podem ser vistas como programas/scripts. Ela permite alterar de forma global o banco de dados.

Com elas, conseguimos utilizar instruções INSERT, UPDATE, DELETE.

Procedures são utilizadas normalmente para juntar várias queries em um único bloco de código e executar alguma tarefa repetitiva ou complexa dentro do banco de dados.

Stored Procedures: (Procedimentos Armazenados)

A sintaxe de uma Procedure é mostrada abaixo:

```
DELIMITER $$
```

```
CREATE PROCEDURE nome_storedprocedure(param1 tipo1, param2 tipo2)
```

```
BEGIN
```

```
    DECLARE var1 tipo1;  
    DECLARE var2 tipo2;
```

```
    instruções1;  
    instruções2;  
    instruções3;
```

```
END $$
```

```
DELIMITER ;
```

Stored Procedures: Exemplo 1

No primeiro exemplo, vamos ver como criar uma Procedure que seja capaz de receber dois parâmetros de entrada (Preço e ID do produto) e atualizar no banco de dados o Preço para o ID do produto informado.

A solução é mostrada ao lado.

Para chamar uma Procedure, usamos a instrução CALL.

```

1   DELIMITER $$ 
2 •  CREATE PROCEDURE sp_AtualizaPreco(NovoPreco DECIMAL(10, 2), ID INT)
3   BEGIN
4       UPDATE dCursos
5           SET Preco_Curso = NovoPreco
6           WHERE ID_Curso = ID;
7       SELECT 'Preço atualizado com sucesso!';
8   END $$ 
9   DELIMITER ;
10
11 •  CALL sp_AtualizaPreco(600, 1);
12
13 •  SELECT * FROM dCursos;
```

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code above is pasted into the editor. Below the code, the results of the query are displayed in a result grid:

ID_Curso	Nome_Curso	Preco_Curso
1	Excel	600.00
2	VBA	200.00
3	Power BI	150.00
*	NULL	NULL

Stored Procedures: Exemplo 2

No próximo exemplo, criamos uma Procedure que aplica um desconto a um determinado curso, de acordo com o seu ID.

A Procedure `sp_AplicaDesconto` recebe dois parâmetros (ID e Desconto) e através de uma sequência de códigos aplica o novo preço do curso dentro da tabela `dCursos`.

```

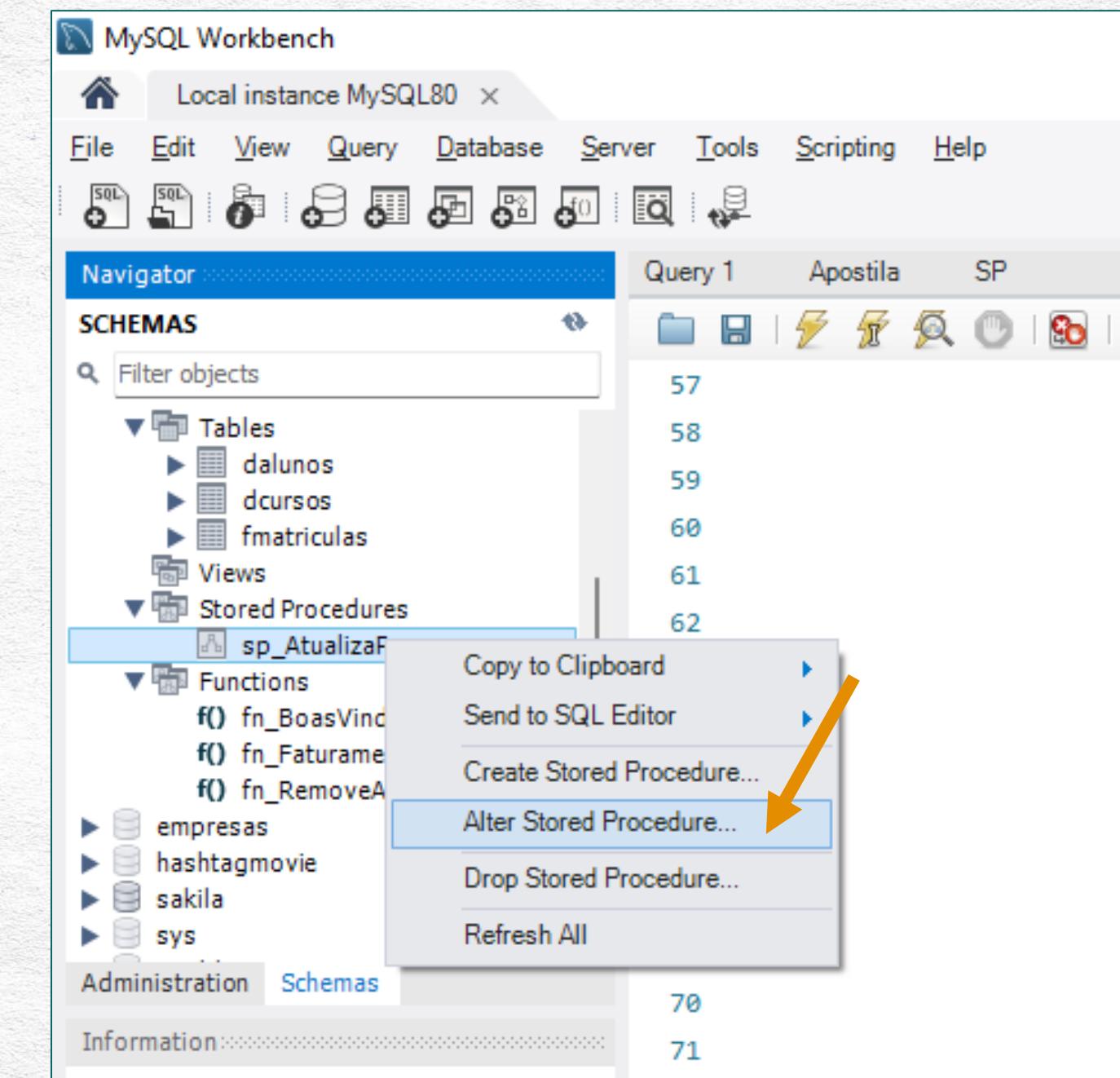
16 ######
17 DELIMITER $$ 
18 • CREATE PROCEDURE sp_AplicaDesconto(ID INT, Desconto DECIMAL(10, 2))
19 • BEGIN
20     DECLARE varPrecoComDesconto DECIMAL(10, 2);
21     DECLARE varNomeCurso VARCHAR(100);
22
23     SET varPrecoComDesconto = (SELECT Preco_Curso FROM dCursos WHERE ID_Curso = ID) * (1 - Desconto);
24     SET varNomeCurso = (SELECT Nome_Curso FROM dCursos WHERE ID_Curso = ID);
25
26     UPDATE dCursos
27     SET Preco_Curso = varPrecoComDesconto
28     WHERE ID_Curso = ID;
29     SELECT 'Desconto aplicado com sucesso!';
30 END $$ 
31 DELIMITER ;
32
33 • SELECT * FROM dCursos;
34
35 • CALL sp_AplicaDesconto(1, 0.5);

```

Result Grid		
ID_Curso	Nome_Curso	Preco_Curso
1	Excel	300.00
2	VBA	200.00
3	Power BI	150.00
NULL	NULL	NULL

Alterando uma Procedure

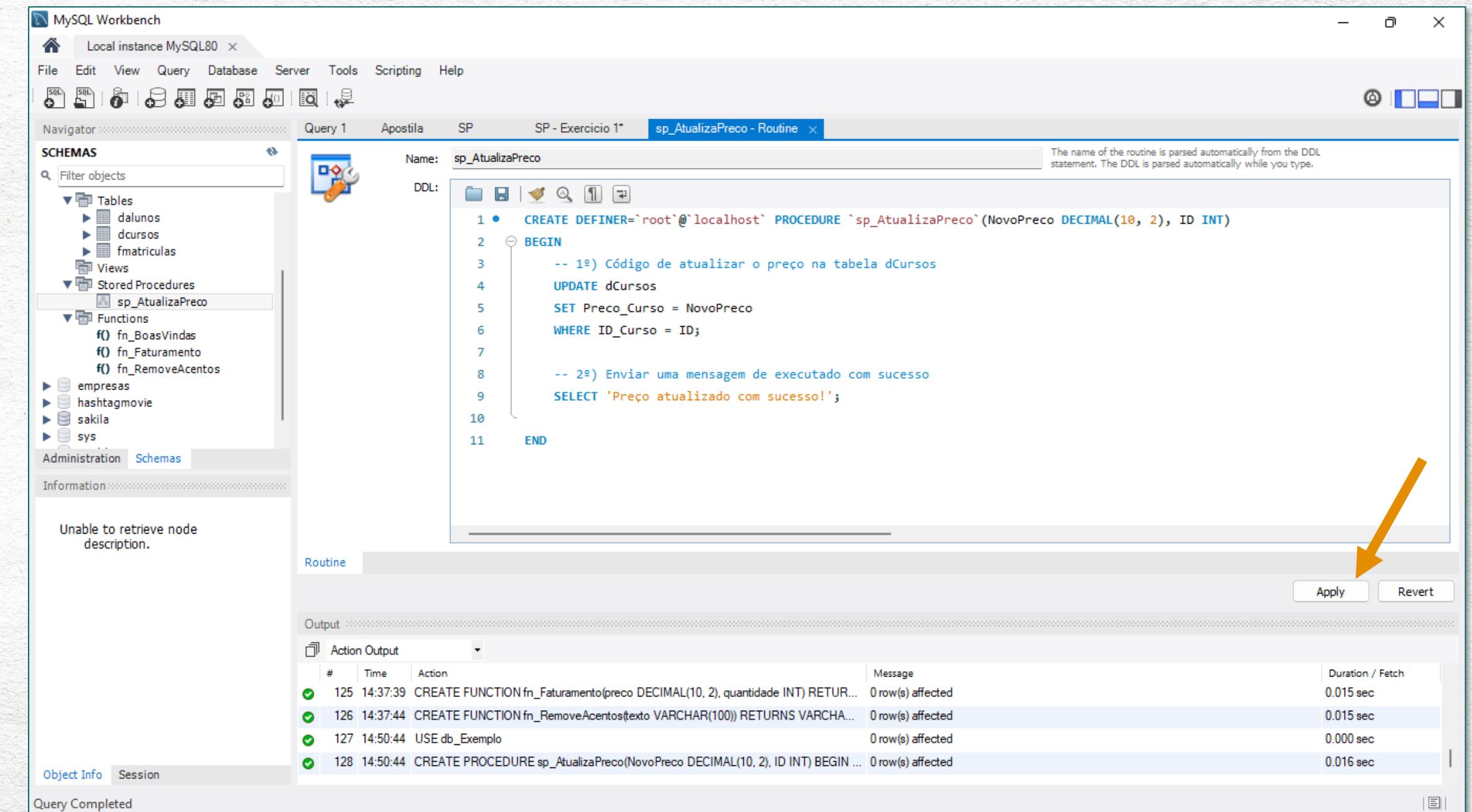
Para alterar uma Procedure, do lado esquerdo, na lista de Procedures criadas, clique com o botão direito na que você deseja alterar e no menu que aparecer, utilize a opção Alter Procedure.



Alterando uma Procedure

Feito isso, uma janela se abrirá onde você pode visualizar a Procedure criada, e consequentemente alterar essa procedure.

Após as alterações, basta clicar em Apply (conforme indicado pela seta ao lado).



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with tables (dalunos, dcursos, fmatriculas), views, and stored procedures. The stored procedure `sp_AtualizaPreco` is selected.
- Query 1:** A tab labeled `Apostila` is active. The tab title is `SP - Exercicio 1*` and the sub-tab is `sp_AtualizaPreco - Routine`.
- DDL:** The code area contains the following SQL:

```

1 • CREATE DEFINER='root'@'localhost' PROCEDURE `sp_AtualizaPreco`(NovoPreco DECIMAL(10, 2), ID INT)
2 BEGIN
3     -- 1º) Código de atualizar o preço na tabela dCursos
4     UPDATE dcursos
5     SET Preco_Curso = NovoPreco
6     WHERE ID_Curso = ID;
7
8     -- 2º) Enviar uma mensagem de executado com sucesso
9     SELECT 'Preço atualizado com sucesso!';
10
11 END

```
- Output:** Shows the execution log with four entries:

#	Time	Action	Message	Duration / Fetch
125	14:37:39	CREATE FUNCTION fn_Faturamento(preco DECIMAL(10, 2), quantidade INT) RETURNS DECIMAL(10, 2)	0 row(s) affected	0.015 sec
126	14:37:44	CREATE FUNCTION fn_RemoveAcentos(texto VARCHAR(100)) RETURNS VARCHAR(100)	0 row(s) affected	0.015 sec
127	14:50:44	USE db_Exemplo	0 row(s) affected	0.000 sec
128	14:50:44	CREATE PROCEDURE sp_AtualizaPreco(NovoPreco DECIMAL(10, 2), ID INT) BEGIN ...	0 row(s) affected	0.016 sec
- Buttons:** At the bottom right, there are `Apply` and `Revert` buttons. An orange arrow points to the `Apply` button.

Excluindo uma Procedure

Para excluir uma Procedure, basta usar o comando `DROP PROCEDURE`, conforme exemplo ao lado.

```
DROP PROCEDURE |ConsultaID|;
```

Functions vs Stored Procedures

Functions

- Uma function é usada para calcular um valor baseado em alguns inputs.
- Functions não permitem a utilização de instruções INSERT, UPDATE e DELETE para alteração de um banco de dados.
- Usamos functions em conjunto com as instruções SELECT, WHERE, HAVING mas não é possível fazer o mesmo com procedures.
- Uma function pode ser executada dentro de uma procedure, mas o contrário não é válido.



Procedures

- Uma procedure pode retornar um ou mais valores, ou não retornar nenhum valor.
- Procedures podem ser vistas como programas/scripts. Uma procedure permite alterar o estado global da base de dados (por exemplo, a utilização das instruções INSERT, UPDATE, DELETE).
- Procedures são utilizadas normalmente para juntar várias queries em um único bloco de comando.

MÓDULO 14

S U B Q U E R I E S

S U B Q U E R I E S

S U B Q U E R I E S



O que é uma subquery?

Uma Subconsulta (ou Subquery ou SubSELECT) nada mais é do que uma consulta dentro de outra consulta. Ou seja, com uma subquery conseguimos utilizar o resultado de uma query (consulta) dentro de outra consulta.

O exemplo abaixo ilustra bem a ideia principal de uma subquery. Para descobrir os produtos que têm o Preco_Unit maior do que a média, podemos adicionar um SELECT dentro de outro SELECT para criar uma consulta mais otimizada.

```
SELECT AVG(Preco_Unit) FROM produtos; -- Média = 1788.125

SELECT
  *
FROM produtos
WHERE Preco_Unit > 1788.125;
```



```
SELECT
  *
FROM produtos
WHERE Preco_Unit > (SELECT AVG(Preco_Unit) FROM produtos);
```

Onde utilizamos subqueries

Subqueries com WHERE (escalar)

Nesta situação, utilizamos o resultado de um SELECT para filtrar um outro SELECT.

Observe no print ao lado. Temos um SELECT mais interno sendo utilizado com o filtro WHERE de uma consulta principal.

Para este caso, utilizamos o termo “escalar” para dizer que o resultado do SELECT mais interno é um valor único, e não uma lista de valores.

```
SELECT
    Coluna1,
    Coluna2
FROM Tabela
WHERE Coluna1 = (SELECT);
```

Onde utilizamos subqueries

Subqueries com WHERE (lista)

Aqui temos uma situação semelhante à anterior, com a diferença de que o resultado da subquery seria uma lista de vários valores que poderiam ser passados no WHERE.

Neste caso, como o filtro seria de mais de 1 valor, utilizamos o IN.

```
SELECT  
    Coluna1,  
    Coluna2  
FROM Tabela  
WHERE Coluna1 IN (SELECT);
```

Onde utilizamos subqueries

Subqueries dentro de um SELECT

Outra opção seria utilizar o resultado de um SELECT como uma nova coluna de uma query principal.

```
SELECT  
    Coluna1,  
    Coluna2,  
    (SELECT)  
FROM Tabela;
```

Onde utilizamos subqueries

Subqueries após um FROM

Podemos utilizar o resultado de um SELECT como uma tabela de um outro SELECT.

Neste caso, precisamos utilizar o comando AS para dar um nome para essa tabela.

```
SELECT
    Coluna1,
    Coluna2
FROM (SELECT) AS T;
```

Subquery como filtro de uma nova consulta (Escalar) - Exemplo 1

No exemplo ao lado, utilizamos uma subconsulta para descobrir quais pedidos foram feitos no banco de dados Base, onde o nome do Gerente é ‘Julia Freitas’.

Como na tabela “pedidos” não temos a informação de nome do gerente (apenas o ID_Loja) usamos uma subquery para descobrir qual é o ID da loja cujo Gerente é a Julia Frentes, para somente então filtrar a tabela Pedidos.

```
-- Exercicio 1. Quais foram os pedidos realizados na loja onde o gerente é o Marcelo Castro?

SELECT * FROM pedidos;
SELECT * FROM lojas;

SET @varNomeGerente = 'Julia Freitas';

SELECT
    *
FROM pedidos
WHERE ID_Loja = (
    SELECT ID_Loja
    FROM lojas
    WHERE Gerente = @varNomeGerente
);
```

Subquery como filtro de uma nova consulta (Escalar) - Exemplo 2

Já no exemplo a seguir, queremos saber quais são os produtos que têm o Preco_Unit maior do que a média.

Para isso, em vez de descobrir primeiro qual é a média para só depois filtrar a tabela, podemos usar o SELECT que calcula essa média diretamente no filtro da tabela de produtos, conforme mostrado ao lado.

```
-- 1. Subquery como filtro de uma nova consulta  
-- Exercício 2. Quais produtos têm o Preco_Unit acima da média?  
  
SELECT AVG(Preco_Unit) FROM produtos; -- Média = 1788.125
```

```
SELECT  
    *  
FROM produtos  
WHERE Preco_Unit > (SELECT AVG(Preco_Unit) FROM produtos);
```

Subquery como filtro de uma nova consulta (Escalar) - Exemplo 3

No próximo exemplo, utilizamos uma subquery para filtrar a tabela de Produtos, para mostrar apenas os produtos que são da Categoria Notebook.

```
-- 1. Subquery como filtro de uma nova consulta
-- Exercicio 3. Quais produtos são da categoria 'Notebook'?

SELECT * FROM categorias;

SELECT * FROM produtos
WHERE ID_Categoria = (
    SELECT
        ID_Categoria
    FROM categorias
    WHERE Categoria = 'Notebook'
);
```

Subquery como filtro de uma nova consulta (Escalar) - Exemplo Desafio

No próximo exemplo, temos um desafio.

Precisamos saber as informações do cliente que gerou mais receita para a empresa. Mas como fazer isso?

```
-- 1. Subquery como filtro de uma nova consulta
-- Exercício Desafio. Descubra todas as informações sobre o cliente que gerou mais receita para a empresa.

SELECT * FROM clientes
WHERE ID_Cliente =
    (
        SELECT
            ID_Cliente
        FROM pedidos
        GROUP BY ID_Cliente
        ORDER BY SUM(Receita_Venda) DESC
        LIMIT 1
    );
```

A consulta mais interna retorna o ID do cliente que, agrupado pela soma de receita de venda, ordenado em ordem crescente e considerando apenas o primeiro (LIMIT 1) nos dá exatamente o ID do cliente que gerou mais receita para a empresa. De posse dessa informação, passamos o resultado dessa consulta mais interna para a consulta mais externa, retornando as informações para o cliente cujo ID é o mesmo retornado pela consulta mais interna (subconsulta).

Subquery como filtro de uma nova consulta (Lista) - Exemplo 1

Agora vamos ver como funcionam as subqueries que retornam listas como resultado.

Ao lado, queremos a soma total de receita, mas apenas para os produtos cuja marca é DELL.

Na tabela Pedidos não temos a informação do nome da marca, apenas o ID do produto. Por isso, usamos uma subquery que retorna todos os Ids da marca DELL, para somente então filtrar a consulta principal.

```
-- Exercicio 1. Descubra qual foi a receita total associada aos produtos da marca DELL.

SELECT
    SUM(Receita_Venda) AS 'Receita Marca DELL'
FROM pedidos
WHERE ID_Produto IN (
    SELECT
        ID_Produto
    FROM produtos
    WHERE Marca_Produto = 'DELL'
);
```

Subquery como filtro de uma nova consulta (Lista) - Exemplo 2

No exemplo ao lado, queremos saber quais pedidos foram feitos na região 'Sudeste'.

Para isso, utilizamos a solução mostrada ao lado.

Observe que para chegar no resultado final, utilizamos duas subqueries: a mais interna para descobrir as cidades associadas à região Sudeste, e uma intermediária para, a partir das Cidades, descobrir os Ids das lojas.

```
-- Exercicio 2. Quais pedidos foram feitos na região 'Sudeste'.  
  
SELECT  
    *  
FROM pedidos  
WHERE ID_Loja IN (  
    SELECT  
        ID_Loja  
    FROM lojas  
    WHERE Loja IN (  
        SELECT  
            Cidade  
        FROM locais  
        WHERE Região = 'Sudeste'  
    )  
);
```

Subquery para criar uma nova coluna na consulta

Nessa nova situação, queremos incluir na consulta uma coluna que retorne a média geral de preços.

Para isso, utilizamos uma subquery dentro do SELECT da consulta mais externa, conforme mostrado ao lado.

```

7 •  SELECT
8      *,
9      (
10         SELECT
11             AVG(Preco_Unit)
12         FROM produtos) AS 'Média Geral de Preço'
13     FROM produtos;

```

	ID_Produto	Nome_Produto	ID_Categoria	Marca_Produto	Num_Serie	Preco_Unit	Custo_Unit	Média Geral de Preço
1	Monitor LED 19,5" Full HD HDMI	1	DELL	MNT-DL-831923	2300	966	1788.1250	
2	Monitor Curvo 24" 144Hz HDMI	1	SAMSUNG	MNT-SS-001939	2800	980	1788.1250	
3	Webcam Full HD 1080p	1	LOGITECH	WBC-LT-934GG4	450	90	1788.1250	
4	Kit Teclado + Mouse sem fio Wireless	2	DELL	KTM-DL-041039	350	129.5	1788.1250	
5	Kit Teclado + Mouse Slim Bluetooth	2	DELL	KTM-DL-111924	280	109.2	1788.1250	
6	Cadeira Gamer reclinável Azul/Laranja	3	ALTURA	CGM-AL-9N914J	1800	540	1788.1250	
7	Cadeira Gamer PC Racer Vermelha	3	ALTURA	CGM-AL-0147FI	3100	1395	1788.1250	
8	Headphone Bluetooth 2000	4	SONY	HDP-SN-194821	600	258	1788.1250	
9	Fone de Ouvido Tune T5000	4	JBL	HDP-JB-091934	780	327.6	1788.1250	
10	Microfone Condensador MC1000	5	AKG	MIC-AK-237591	1100	275	1788.1250	

Subquery para criar uma tabela na consulta

A última aplicação da subconsulta é como uma tabela dentro da consulta principal.

No exemplo ao lado, queremos saber qual foi a quantidade máxima vendida, a quantidade mínima vendida e a média de vendas.

Para isso, criamos uma subconsulta mais interna, que retorna o agrupamento das vendas por produto. Como essa query retorna uma tabela, alimentamos a tabela mais externa com o resultado dessa query mais interna, e calculamos o máximo, mínimo e média de vendas, conforme mostrado ao lado.

```
-- Exercício 1. Do total de vendas por produto, qual foi a quantidade máxima vendida? E a quantidade mínima? E a média?

SELECT
    MAX(Vendas) AS 'Máximo Vendas',
    MIN(Vendas) AS 'Mínimo Vendas',
    AVG(Vendas) AS 'Média Vendas'
FROM (
    SELECT
        ID_Produto,
        COUNT(*) AS 'Vendas'
    FROM pedidos
    GROUP BY ID_Produto
) AS t;
```

	Máximo Vendas	Mínimo Vendas	Média Vendas
349	6	93.5000	

EXISTS

O operador EXISTS é usado para testar a existência de qualquer registro em uma subconsulta. Ele retorna TRUE se a subconsulta retornar um ou mais registros.

A sintaxe é mostrada abaixo:

SELECT coluna(s)

FROM tabela

WHERE EXISTS (SELECT);

EXISTS (Exemplo)

Na imagem abaixo, temos um exemplo de aplicação do EXISTS.

A ideia é verificar se todas as categorias possuem pelo menos 1 exemplar de produtos.

-- Exercício. Você deverá verificar se todos as categorias possuem pelo menos 1 exemplar de produtos (na tabela de produtos); caso alguma categoria não possua nenhum exemplar você deverá descobrir qual é/quais são.

EXISTS (Exemplo)

Para resolver essa questão, usamos o EXISTS como mostrado ao lado.

O retorno do EXISTS será uma tabela com todas as categorias que possuem pelo menos um exemplar de produtos.

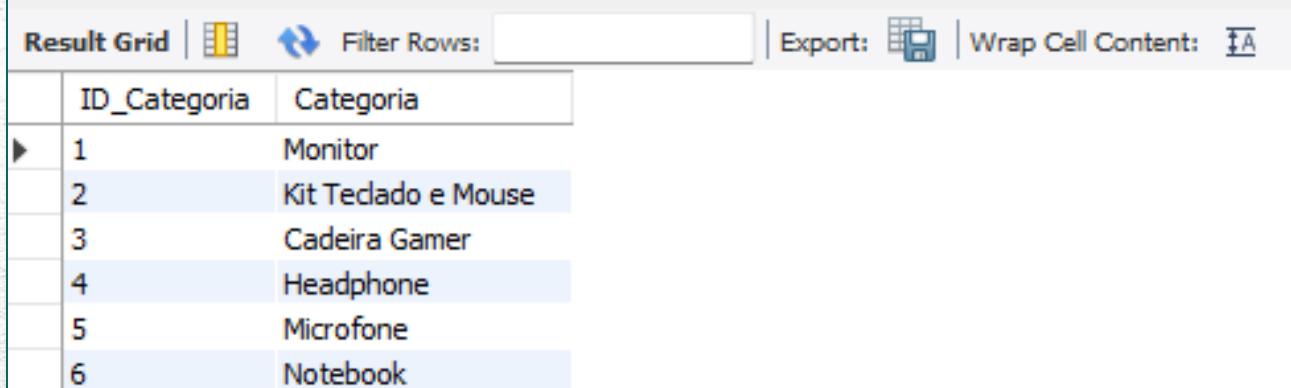
E como fazemos para descobrir qual categoria não tem um exemplar?

Podemos usar o NOT EXISTS.

```

19 • SELECT *
20   FROM categorias
21 WHERE EXISTS (
22   SELECT *
23     FROM produtos
24   WHERE categorias.ID_Categoria = produtos.ID_Categoria
25 );
26
27
28
29

```



The screenshot shows the MySQL Workbench interface with the SQL editor containing the provided code. Below the editor is a result grid titled "Result Grid". The grid has two columns: "ID_Categoria" and "Categoria". Six rows are displayed, corresponding to the categories listed in the code: Monitor, Kit Teclado e Mouse, Cadeira Gamer, Headphone, Microfone, and Notebook. The second row, "Kit Teclado e Mouse", is highlighted with a blue background.

	ID_Categoria	Categoria
▶	1	Monitor
	2	Kit Teclado e Mouse
	3	Cadeira Gamer
	4	Headphone
	5	Microfone
	6	Notebook

EXISTS (Exemplo)

Com o NOT EXISTS, verificamos a não existência da informação.

Ao fazer essa pequena modificação no código, o resultado é exatamente a categoria que não possui nenhum exemplar de produto: no caso, a categoria Acessórios.

```
19 • SELECT
20      *
21  FROM categorias
22 WHERE NOT EXISTS (
23             SELECT
24                 *
25             FROM produtos
26             WHERE categorias.ID_Categoria = produtos.ID_Categoria
27         );
28
29
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
ID_Categoria	Categoria			
7	Acessórios			

SQL IMPRESSIONADOR

Apostila completa