

Banco de Dados com MySQL - parte 01

O que veremos por aqui:

1. Introdução ao MySQL
2. Entendendo: Comandos e Conceitos Básicos

1. MySQL

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada - Structured Query Language) como interface. É atualmente um dos Sistemas de Gerenciamento de Banco de dados mais populares da Oracle Corporation, com mais de 10 milhões de instalações pelo mundo. Possui versões pagas e a versão Community (gratuita).



[Site Oficial: MySQL](https://www.mysql.com/)



DICA: *Caso você tenha alguma dúvida quanto a instalação do MySQL, consulte o Guia de Instalação do MySQL.

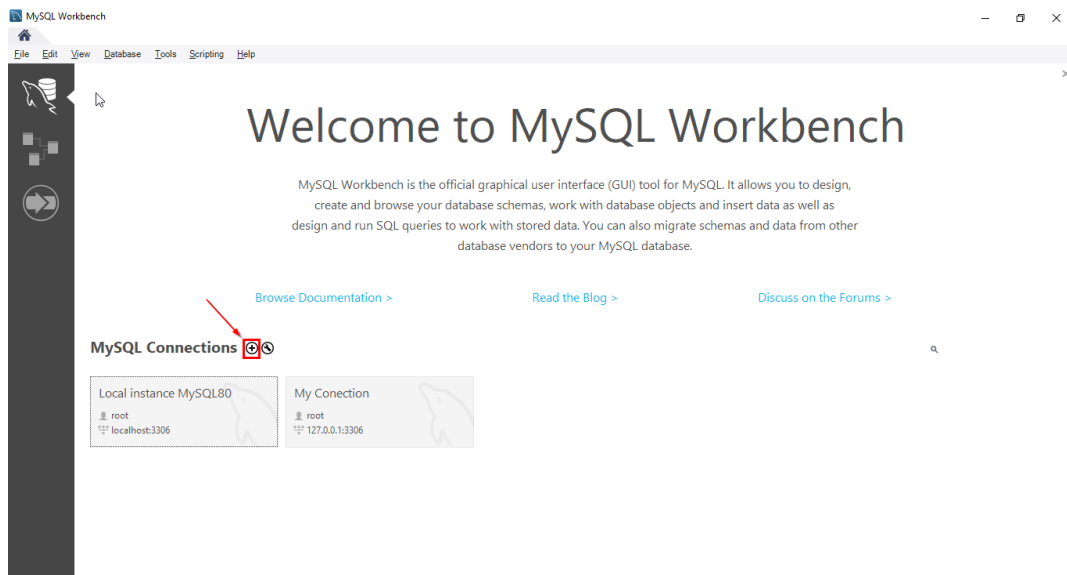
1.1. Entendendo: Comandos e Conceitos Básicos

A IDE que utilizaremos para criar e manipular o nosso banco de dados será o MySQL Workbench

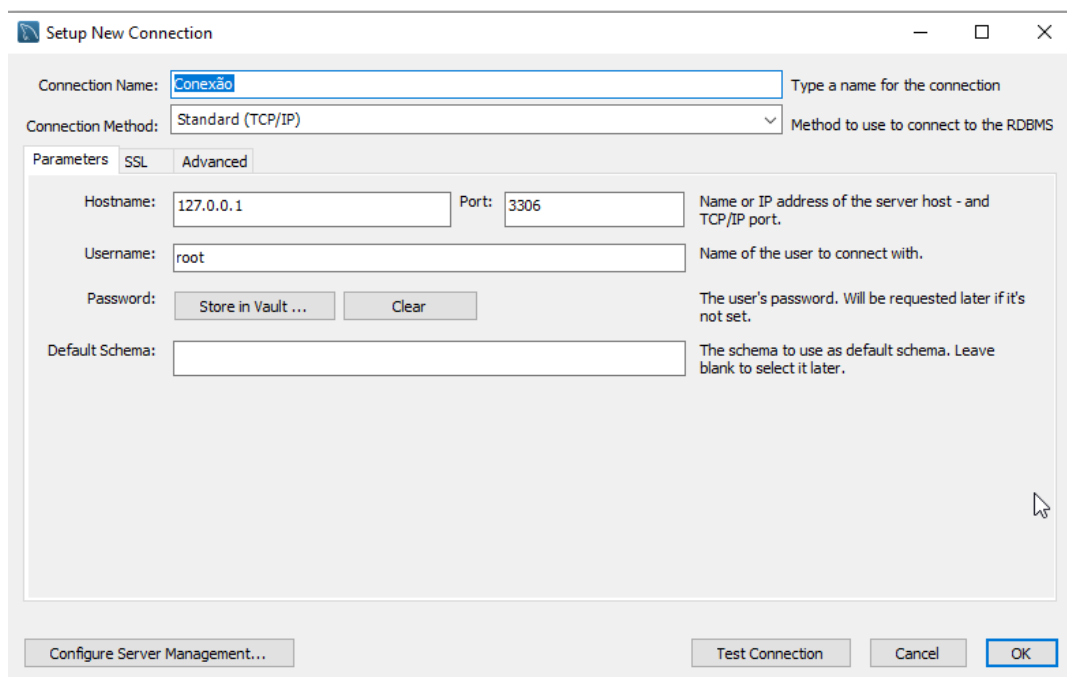


1.2. Criando nosso primeiro Banco de Dados

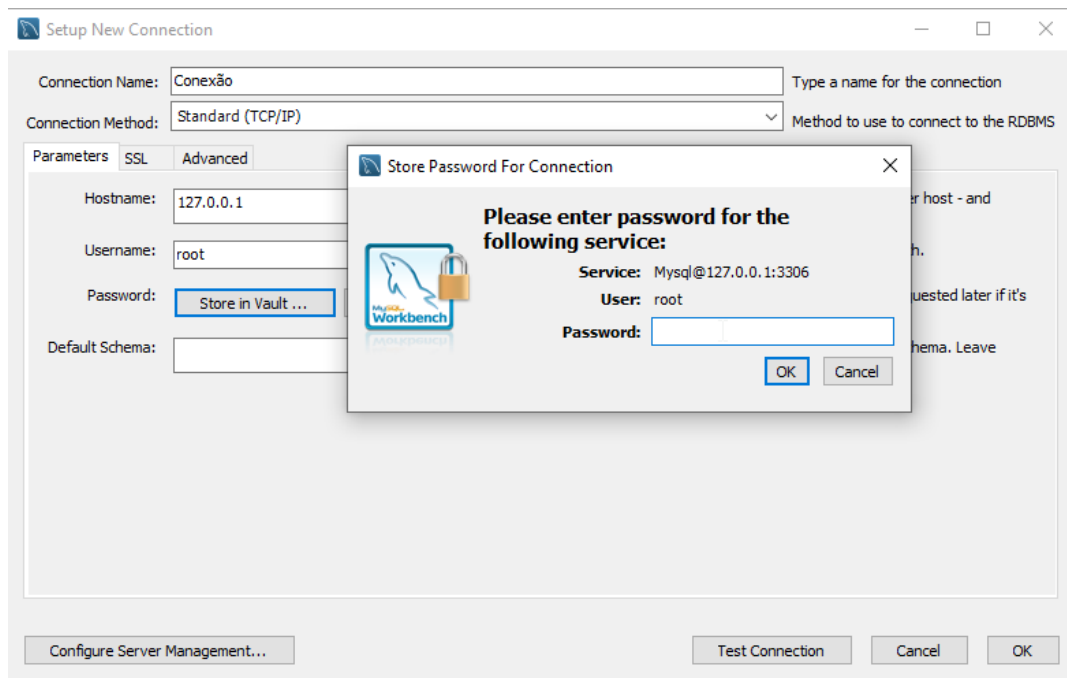
1. Uma vez que estamos o MySQL Workbench aberto, precisamos criar uma conexão local. Portanto, **clique no item com símbolo de “+”, ao lado das palavras **MySQL Connections****.



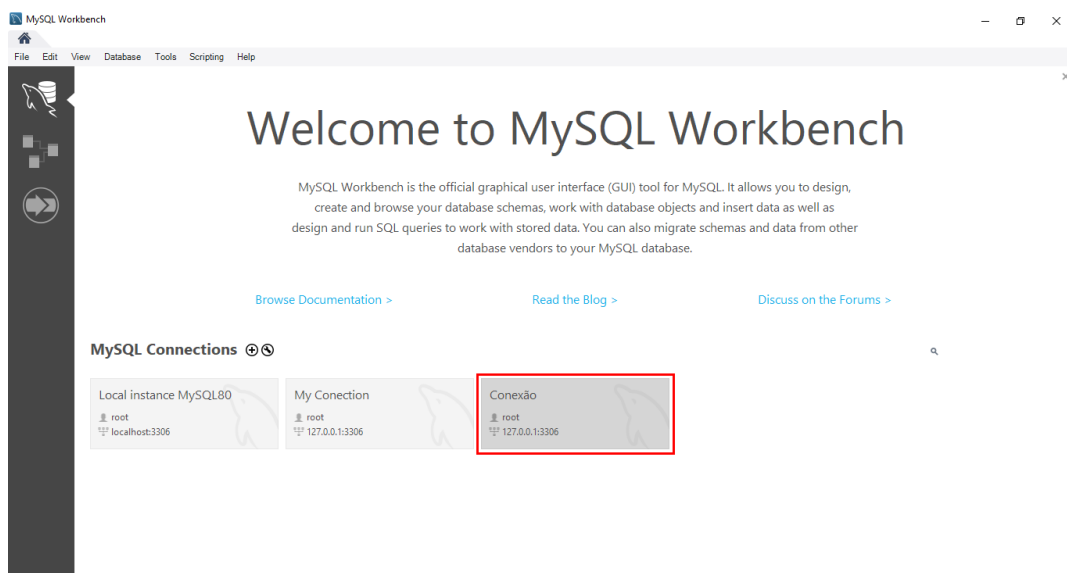
2. Defina um nome para a sua conexão.



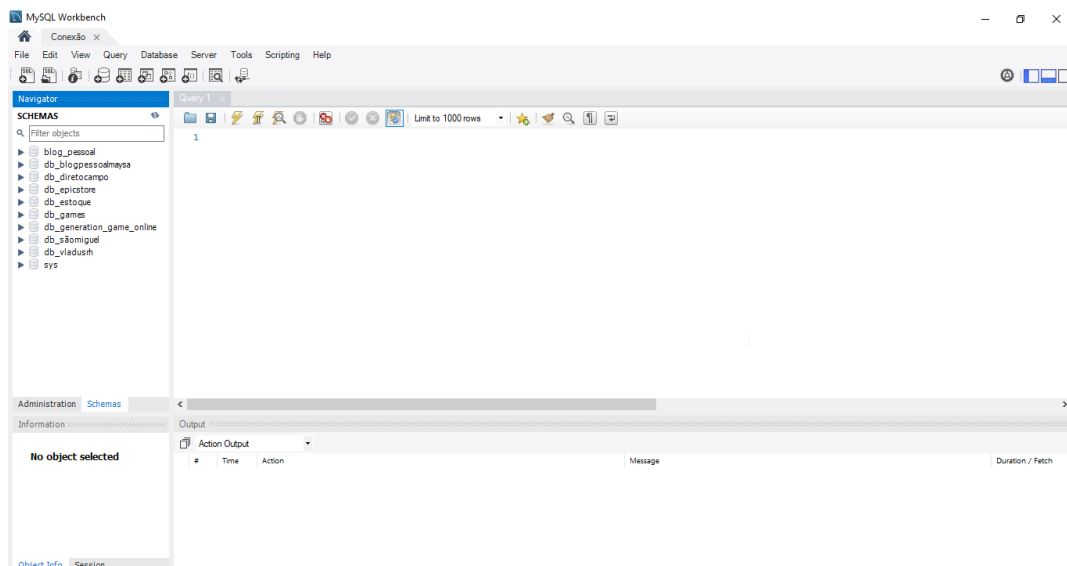
3. Defina uma senha para esta conexão clicando em **Store in Vault...**, e clique em OK.



4. Uma vez feito os passos acima, já temos nossa conexão criada.





5. Após acessar a conexão criada teremos um ambiente onde vamos criar nossas tabelas e relaciona-las.



1.3. Executando consultas no MySQL Workbench


Para escrever o código SQL das nossas Queries, utilize a aba **Query** do MySQL Workbench.

Para executar a Query existem duas maneiras:

	Descrição
	Executa todas as linhas ou apenas as linhas selecionadas de uma vez só na sequência.
	Executa apenas a linha onde o cursor do mouse está posicionado ou apenas a primeira linha de uma seleção.

1.4. Banco de Dados - Exemplo


Utilizaremos como Banco de dados guia para testarmos as SQL Queries no MySQL o Banco **db_quitanda**. Primeiro vamos criar a tabela **tb_produtos**, que possui estrutura abaixo:



tb_produtos	
id	BIGINT UNSIGNED AUTO INCREMENT
nome	VARCHAR(255)
quantidade	INT
preco	DECIMAL(10,0)
Indexes	



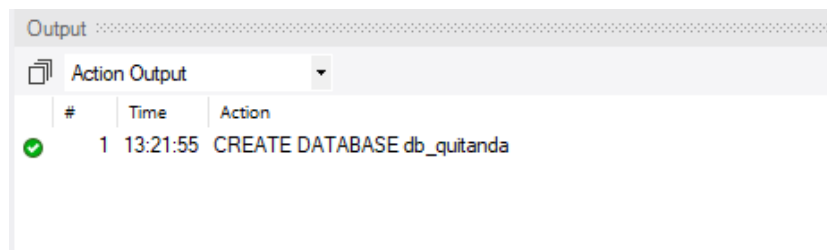
Criar o Banco de dados

Na aba **Query**, digite e execute a query abaixo utilizando o segundo raio  :

CREATE DATABASE

```
CREATE DATABASE db_quitanda;
```

Ao executar a query, será criado o **Banco de Dados** e no campo de Output aparecerá o log/mensagem informando que a operação foi efetuada com sucesso.




Output		
Action Output		
#	Time	Action
✓ 1	13:21:55	CREATE DATABASE db_quitanda



[Documentação: Create database - W3Schools](#)



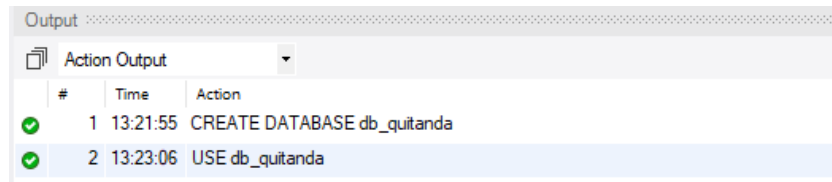
Selecionando o Banco de Dados

Antes de criarmos nossa primeira tabela, é necessário indicar ao MySQL qual banco de dados queremos trabalhar. Para isso utilize a query abaixo, e em seguida execute a query abaixo utilizando o segundo raio .

USE

```
USE db_quitanda;
```

Ao executar a query, aparecerá no log/mensagem do campo de Output a seguinte mensagem:




#	Time	Action
✓ 1	13:21:55	CREATE DATABASE db_quitanda
✓ 2	13:23:06	USE db_quitanda



[Documentação: Use - MySQL Tutorial](#)



Criando nossa primeira Tabela

O processo para criação de uma tabela é bem simples, basta inserir a query abaixo, e em seguida execute a query abaixo utilizando o segundo raio .

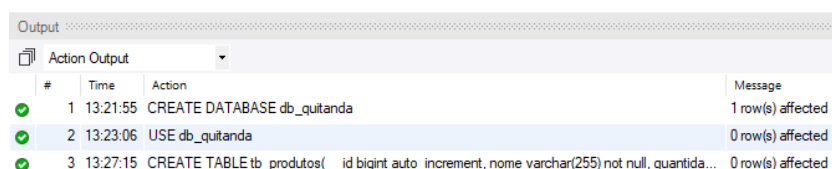
CREATE TABLE

```
CREATE TABLE tb_produtos(  
  id bigint auto_increment,  
  nome varchar(255) not null,  
  quantidade int,  
  preco decimal not null,  
  PRIMARY KEY (id)  
);
```

Caso a query tenha sido executada corretamente, no canto esquerdo na aba das **Tabelas** aparecerá a tabela criada:



E aparecerá no log/mensagem do campo de Output a seguinte mensagem:



#	Time	Action	Message
✓ 1	13:21:55	CREATE DATABASE db_quitanda	1 row(s) affected
✓ 2	13:23:06	USE db_quitanda	0 row(s) affected
✓ 3	13:27:15	CREATE TABLE tb_produtos(id bigint auto_increment, nome varchar(255) not null, quantida...	0 row(s) affected



[Documentação: Create Table - W3Schools](#)



[Documentação: Primary Key - W3Schools](#)



Inserindo dados na Tabela

Uma vez que já temos a tabela criada, precisamos popula-la, ou seja, inserir dados. Para tal digite o código abaixo seguindo a sintaxe/estrutura de código abaixo e em seguida selecione as 6 linhas e execute a query utilizando o primeiro raio ⚡.

INSERT INTO

```
INSERT INTO tb_produtos(nome, quantidade, preco)
values ("tomate",100, 8.00);
INSERT INTO tb_produtos(nome, quantidade, preco)
values ("maçã",20, 5.00);
INSERT INTO tb_produtos(nome, quantidade, preco)
values ("laranja",50, 10.00);
INSERT INTO tb_produtos(nome, quantidade, preco)
values ("banana",200, 12.00);
INSERT INTO tb_produtos(nome, quantidade, preco)
values ("uva",1200, 30.00);
INSERT INTO tb_produtos(nome, quantidade, preco)
values ("pêra",500, 2.99);
```

Feito isto os dados dos produtos Tomate, Maçã, Laranja e das outras frutas serão inseridos na tabela.



IMPORTANTE: Como definimos que o campo `id` será chave primária (PRIMARY KEY) e será atualizado automaticamente de um em um a cada registro inserido, através do código `auto_increment`, não inserimos nenhum valor nesse campo.



[Documentação: Insert Into - W3Schools](#)



Selecionando dados da Tabela

Agora vamos fazer a nossa primeira consulta no bando para ver se os dados foram devidamente inseridos na tabela. Para isto basta fazer um `select` na tabela:

SELECT

```
SELECT * FROM tb_produtos;
```

Ao executar a query, aparecerá o console com a seguinte tabela:

Result Grid				
Filter Rows:				
	id	nome	quantidade	preco
▶	1	tomate	100	8
	2	maçã	20	5
	3	laranja	50	10
	4	banana	200	12
	5	uva	1200	30
	6	pêra	500	3
*	NULL	NULL	NULL	NULL

Caso seja necessário selecionar apenas um atributo como por exemplo o nome, basta substituir o (*) **asterisco** pelo nome do atributo desejado.

```
SELECT nome FROM tb_produtos;
```

Ao executar a query, aparecerá o console com a seguinte tabela:

Result Grid	
Filter Rows:	
	nome
▶	tomate
	maçã
	laranja
	banana
	uva
	pêra

Caso seja necessário mostrar mais de um atributo, **devemos separa-los por vírgula**.

```
SELECT nome, preco FROM tb_produtos;
```

Ao executar a query, aparecerá no log/mensagem do campo de Output a seguinte tabela:

Result Grid		
Filter Rows:		
	nome	preco
▶	tomate	8
	maçã	5
	laranja	10
	banana	12
	uva	30
	pêra	3



[Documentação: Select - W3Schools](#)



Selecionando dados da Tabela com critérios

Caso haja a necessidade de retornar apenas uma linha especifica, utilizarmos a cláusula `WHERE` seguido do atributo que queremos filtrar. **Exemplo:** desejamos retornar todas as informações do Produto cujo `id` é igual 1. Nesse caso executamos a seguinte query:

WHERE

```
SELECT * FROM tb_produtos WHERE id = 1;
```

Ao executar a query, teremos o seguinte resultado:

Result Grid

Filter Rows:

Edit:

	id	nome	quantidade	preco
▶	1	tomate	100	8
✱	NULL	NULL	NULL	NULL



[Documentação: Where - W3Schools](#)



Selecionando dados com os Operadores Relacionais

Caso haja a necessidade de retornar registro com base em comparações, basta utilizarmos o código `WHERE` junto com os Operadores Relacionais, informando o campo/coluna que será utilizado na comparação.

Operador	Descrição
>	Maior do que
>=	Maior do que ou igual
<	Menor do que
<=	Menor do que ou igual
=	igual
<>	diferente

Exemplo: desejamos retornar todos os produtos que tenham preço acima de R\$ 5,00. Nesse caso executamos a seguinte query:

WHERE com Operados Relacionais

```
SELECT * FROM tb_produtos WHERE preco > 5.00;
```

Ao executar a query, teremos o seguinte resultado:

Result Grid

Filter Rows:

Edit:

	id	nome	quantidade	preco
▶	1	tomate	100	8
	3	laranja	50	10
	4	banana	200	12
	5	uva	1200	30
✱	NULL	NULL	NULL	NULL



[Documentação: Operadores - W3Schools](#)



Selecionando dados com os Operadores Lógicos

Caso haja a necessidade de retornar um registro com base em um resultado lógico (VERDADEIRO ou FALSO), basta utilizarmos o código `WHERE` junto com os Operadores Lógicos, informando o atributo que será utilizado na comparação.

Operador	Descrição
AND	Verdadeiro somente se todas as condições forem verdadeiras
OR	Verdadeiro se 1 condição for verdadeira
NOT	Negação

Exemplo: desejamos retornar todos os produtos que tenham preço acima de R\$5,00 E quantidade menor do que 100. Nesse caso executamos a seguinte query:

WHERE com Operadores Lógicos

```
SELECT * FROM tb_produtos WHERE preco > 5.00 AND quantidade < 100;
```

Ao executar a query, teremos o seguinte resultado:

Result Grid				
Filter Rows: <input type="text"/>				
	id	nome	quantidade	preco
▶	3	laranja	50	10
✱	NULL	NULL	NULL	NULL



[Documentação: Operadores Relacionais- W3Schools](#)



Atualizando dados na Tabela

Para atualizar os dados da tabela devemos utilizar o comando `update` no registro que queremos atualizar. **Exemplo:** queremos alterar o preço do produto Tomate para R\$5,00.



ALERTA DE BSM: Mantenha a Atenção aos Detalhes ao construir essa Query, pois obrigatoriamente você DEVE COLOCAR o comando WHERE, pois caso contrário o MySQL Workbench irá atualizar TODOS OS REGISTROS, sem a possibilidade de desfazer a atualização errônea de maneira simples.

Antes de Executar a query, execute o comando abaixo:

```
SET SQL_SAFE_UPDATES = 0;
```

A linha de comando acima **desabilita o modo de atualização segura** do MySQL, ou seja, é um modo que impede, por exemplo, de executar os famosos **UPDATE sem WHERE** (modifica todos o registros da tabela) e o **DELETE sem WHERE** (apaga todos o registros da tabela).

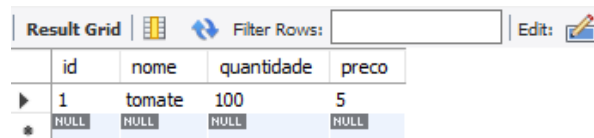
UPDATE

```
UPDATE tb_produtos SET preco = 5.00 WHERE id = 1;
```

Ao executar a query, aparecerá no log/mensagem do campo de Output uma mensagem que a alteração foi concluída, e para visualizar o produto alterado basta executar a seguinte query:

```
SELECT * FROM tb_produtos WHERE id = 1;
```

Logo você terá a seguinte imagem:



The screenshot shows the MySQL Workbench interface with the 'Result Grid' tab selected. It displays a single row of data for the product with id 1. The columns are 'id', 'nome', 'quantidade', and 'preco'. The values are 1, 'tomate', 100, and 5.00 respectively. There is also a row for NULL values.

	id	nome	quantidade	preco
▶	1	tomate	100	5
*	NULL	NULL	NULL	NULL



[Documentação: Update - W3Schools](#)



Apagando dados na Tabela

Para apagar um ou mais registros da tabela, utilizamos o comando `DELETE`. **Exemplo:** queremos apagar o produto que tenha o id igual a 2.



ALERTA DE BSM: Mantenha a Atenção aos Detalhes ao construir essa Query, pois obrigatoriamente você DEVE COLOCAR o comando WHERE, pois caso contrário o MySQL Workbench irá APAGAR TODOS OS REGISTROS, sem a possibilidade de desfazer a operação.

Nesse caso executamos a seguinte query:

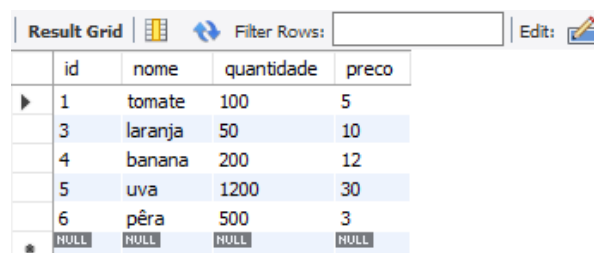
DELETE

```
DELETE FROM tb_produtos WHERE id = 2;
```

Ao executar a query, aparecerá no log/mensagem do campo de Output uma mensagem que a exclusão foi concluída, e para visualizar se o produto foi realmente excluído basta executar a seguinte query para listar os produtos:

```
SELECT * FROM tb_produtos;
```

Logo você terá a seguinte imagem:



The screenshot shows the MySQL Workbench interface with the 'Result Grid' tab selected. It displays a list of products. The columns are 'id', 'nome', 'quantidade', and 'preco'. The rows are: 1 (tomate, 100, 5), 3 (laranja, 50, 10), 4 (banana, 200, 12), 5 (uva, 1200, 30), 6 (pêra, 500, 3). There is also a row for NULL values.

	id	nome	quantidade	preco
▶	1	tomate	100	5
	3	laranja	50	10
	4	banana	200	12
	5	uva	1200	30
	6	pêra	500	3
*	NULL	NULL	NULL	NULL



[Documentação: Delete - W3Schools](#)



Modificando a Estrutura da Tabela

Esse comando é usado para adicionar, excluir ou modificar as colunas (atributos), em uma tabela existente, além de poder adicionar ou excluir restrições de uma tabela, como chaves primárias e/ou chaves estrangeiras.

Exemplo 01: Queremos modificar/atualizar o tipo de um Atributo/Coluna, fazendo com que o campo preco tenha 6 dígitos, sendo que 2 deles são casas decimais (1000.50). Nesse caso executamos a seguinte query:

ALTER TABLE - MODIFY

```
ALTER TABLE tb_produtos MODIFY preco decimal(6,2);
```

Observe que após esta alteração o atributo preco será exibido com as casas decimais:

```
SELECT * FROM tb_produtos;
```

	id	nome	quantidade	preco
▶	1	tomate	100	8.00
	3	laranja	50	10.00
	4	banana	200	12.00
	5	uva	1200	30.00
	6	pêra	500	3.00
*	NULL	NULL	NULL	NULL



DESAFIO: Observe que mesmo corrigindo as casas decimais, o preço da pêra continua arredondado para R\$ 3.00. Como podemos corrigir o valor deste produto?

Exemplo 02: Queremos adiciona um novo Atributo/Coluna na Tabela. Nesse caso executamos a seguinte query:

ALTER TABLE - ADD

```
ALTER TABLE tb_produtos ADD descricao varchar(255);
```

Exemplo 03: Queremos remover um Atributo/Coluna da tabela. Nesse caso executamos a seguinte query:

ALTER TABLE - DROP

```
ALTER TABLE tb_produtos DROP descricao;
```

Exemplo 04: Queremos alterar o nome de um Atributo/Coluna da tabela. Nesse caso executamos a seguinte query:

ALTER TABLE - CHANGE

```
ALTER TABLE tb_produtos CHANGE nome nomeproduto VARCHAR(255);
```



DICA: *Após executar as consultas, execute um `select * from tb_produtos;` e veja as alterações.*



[Documentação: Alter Table - W3Schools](#)