

# Banco de Dados com MySQL - parte 02

O que veremos por aqui:

1. Consultas SQL avançadas
2. Entendendo: Associações entre Tabelas

## 1. Consultas SQL avançadas



### Selecionando dados e ordenando

Também podemos ordenar os registros de acordo com um campo desejado. Caso queira ordenar por nome, basta adicionar o comando `ORDER BY` e o campo que queira ordenar, que no caso seria o campo nome. Em seguida por adicione o código `ASC` para ordenar de forma crescente ou `DESC` para ordenar de forma decrescente.

#### SELECT com ORDER BY

```
SELECT * FROM tb_produtos ORDER BY nome ASC;
```

Ao executar a query, aparecerá no log/mensagem do campo de Output todos os dados da tabela, mas ordenados pelo nome do produto:

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



[Documentação: Order By - W3Schools](#)



### Selecionando dados dentro de um intervalo

Quando desejamos procurar valores que estão dentro de um determinado valor, utilizamos o operador `BETWEEN`. Esses valores podem ser números, texto ou datas. **Exemplo:** queremos retornar todos os produtos que tem o seu preço **ENTRE** R\$3,00 e R\$12,00. Nesse caso executamos a seguinte query:

#### SELECT com o operador BETWEEN

```
SELECT * FROM tb_produtos WHERE preco BETWEEN 3.00 AND 12.00;
```

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

Result Grid

Filter Rows:

Edit:

	id	nome	quantidade	preco
▶	1	tomate	100	8
	2	maçã	20	5
	3	laranja	50	10
	4	banana	200	12
	6	pêra	500	3
✱	NULL	NULL	NULL	NULL



[Documentação: Between - W3Schools](#)



## Selecionando dados dentro de uma lista de critérios



Esse operador permite que retornem vários registros utilizando mais de uma condição de procura no comando `WHERE`, funcionando como um atalho caso tenhamos multiplicas condições e não desejamos utilizar o **Operador OR** dentro do `where`. **Exemplo:** queremos retornar todos os produtos em que seu preço seja R\$3,00, R\$10,00 ou R\$12,00. Nesse caso executamos a seguinte query:

### SELECT com o operador IN

```
SELECT * FROM tb_produtos WHERE preco IN (3.00, 10.00, 12.00);
```


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

Result Grid

Filter Rows:

Edit:



	id	nome	quantidade	preco
▶	3	laranja	50	10
	4	banana	200	12
	6	pêra	500	3
✱	NULL	NULL	NULL	NULL



[Documentação: In - W3Schools](#)



## Selecionando dados que contenham um texto específico (Busca textual)

Quando desejamos procurar um padrão especificado em um registro de uma coluna, usamos o comando `LIKE`. Esse comando é utilizado junto com os caracteres "**coringas/wildcards**": (%) e (\_).

Padrão	Descrição
<b>%texto</b>	O conteúdo procurado deve terminar com o texto digitado (texto)
<b>texto%</b>	O conteúdo procurado deve começar com o texto digitado (texto)
<b>%texto%</b>	O conteúdo procurado deve conter o texto digitado (texto) em qualquer posição.

**Exemplo:** queremos retornar todos os produtos que comecem a letra "L". Nesse caso executamos a seguinte query:

## SELECT com o operador LIKE

```
SELECT * FROM tb_produtos WHERE nome LIKE "l%";
```

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

Result Grid

Filter Rows:

Edit:

	id	nome	quantidade	preco
▶	3	laranja	50	10
✱	NULL	NULL	NULL	NULL



**IMPORTANTE:** A estratégia para retornar qualquer produto que entre seu nome tenha um determinado texto, seja no começo ou no fim ( "%texto%" ), é a mais utilizada.

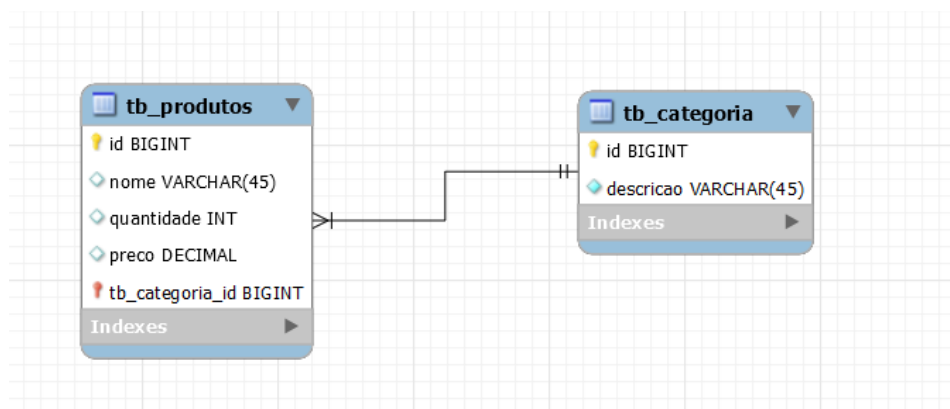


[Documentação: Like - W3Schools](#)

## 2. Entendendo: Associações entre Tabelas

Como vimos no capítulo **Entendendo: Relacionamento entre Tabelas**, podemos definir relacionamentos entre tabelas de 1:1, 1:N e N:N, mas podemos precisar fazer buscas relacionadas entre estas tabelas, para isso precisamos de um mecanismo de busca de Associações entre Tabelas.

Vamos criar a tabela **tb\_categoria** e relacioná-la com a tabela **tb\_produtos**. Será um relacionamento do tipo 1:N, onde uma Categoria pode ter N produtos e N Produtos podem ter apenas 1 Categoria.



1. Informe ao MySQL que iremos usar o Banco de Dados `db_quitanda` através do comando `USE`. Feito isso, crie uma tabela com o nome de `tb_categoria`.

```
USE db_quitanda;
```

```
CREATE TABLE tb_categoria(  
    id bigint auto_increment,  
    descricao varchar(255) not null,  
    primary key (id)  
);
```

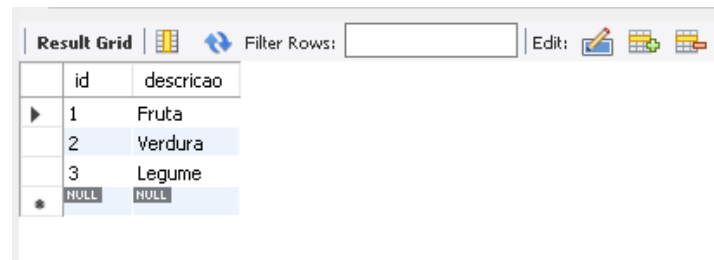
2. Insira os seguintes registros na `tb_categoria`:

```
INSERT INTO tb_categoria (descricao) values ("Fruta");
INSERT INTO tb_categoria (descricao) values ("Verdura");
INSERT INTO tb_categoria (descricao) values ("Legume");
```

3. Para verificar como a tabela está alimentada/populada, execute a seguinte query:

```
SELECT * FROM tb_categoria;
```

4. Você deve ter um resultado como esse:



	id	descricao
▶	1	Fruta
	2	Verdura
	3	Legume
*	NULL	NULL

5. Insira na tabela `tb_produtos` a coluna `categoria_id` com a constraint/restrição Chave Estrangeira/Foreign Key:

```
-- Adicionando a nova coluna
ALTER TABLE tb_produtos ADD categoria_id bigint;

-- Adicionando a constraint
ALTER TABLE tb_produtos ADD CONSTRAINT fk_produtos_categorias
FOREIGN KEY (categoria_id) REFERENCES tb_categoria (id);
```



[Documentação: Foreign key - W3Schools](#)

6. Insira os seguintes registros na `tb_produtos`:

```
INSERT INTO tb_produtos(
    nome, quantidade, preco, categoria_id
) values("Pitaya", 10, 15.00, 1);

INSERT INTO tb_produtos(
    nome, quantidade, preco, categoria_id
) values("Agrião", 15, 3.00, 2);

INSERT INTO tb_produtos(
    nome, quantidade, preco, categoria_id
) values("Cenoura", 18, 3.50, 3);
```

7. Para verificar como a tabela está alimentada/populada, execute a seguinte query:

```
SELECT * FROM tb_produtos;
```

8. Você deve ter um resultado como esse:

SQL File 3"

Limit to 1000 rows

1 • **SELECT \* FROM tb\_produtos;**

Result Grid

Filter Rows:

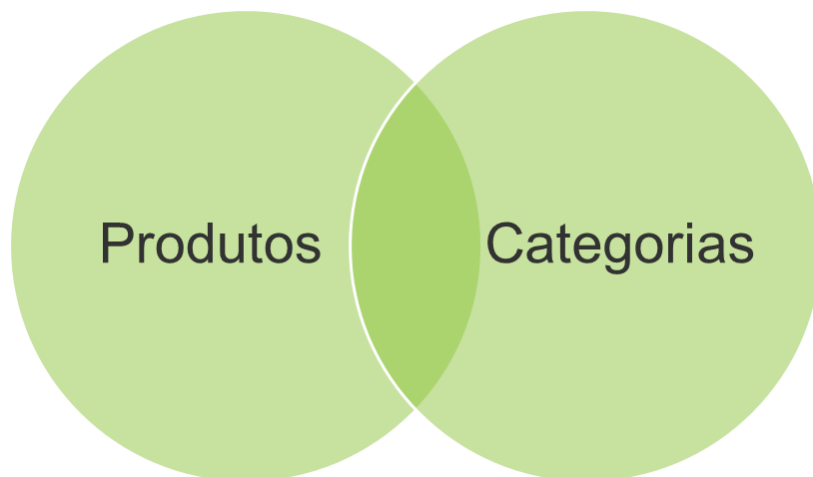
Edit: Export/Import: Wrap Cell Content:

	id	nome	quantidade	preco	categoria_id
▶	1	tomate	100	5	NULL
	3	laranja	50	10	NULL
	4	banana	200	12	NULL
	5	uva	1200	30	NULL
	6	pêra	500	3	NULL
	7	Pitaya	10	15	1
	8	Agrião	15	3	2
	9	Cenoura	18	4	3
*	NULL	NULL	NULL	NULL	NULL

## 2. Associações entre tabelas

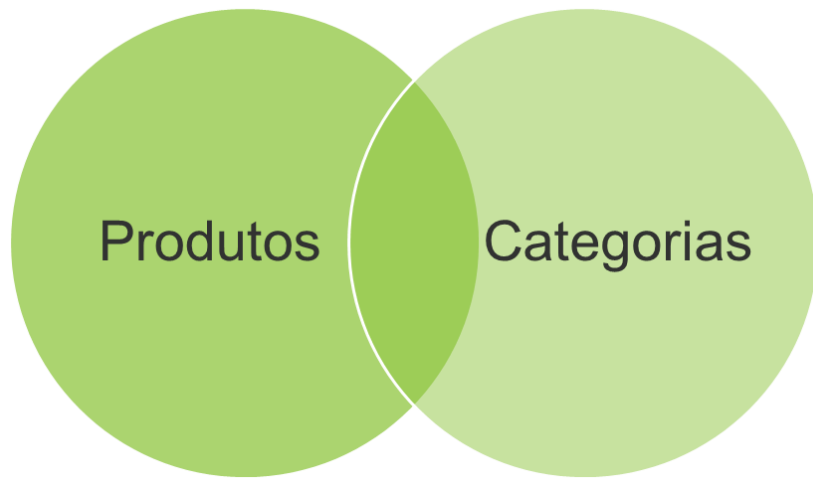
Associações de tabelas ou busca por **JOIN** podem ser utilizadas para diversas finalidades, como converter em informação os dados encontrados em duas ou mais tabelas. A cláusula **JOIN** é usada para combinar dados provenientes de duas ou mais tabelas do banco de dados, baseado em um relacionamento entre colunas destas tabelas. Há três categorias principais de **JOINS**:

### INNER JOIN



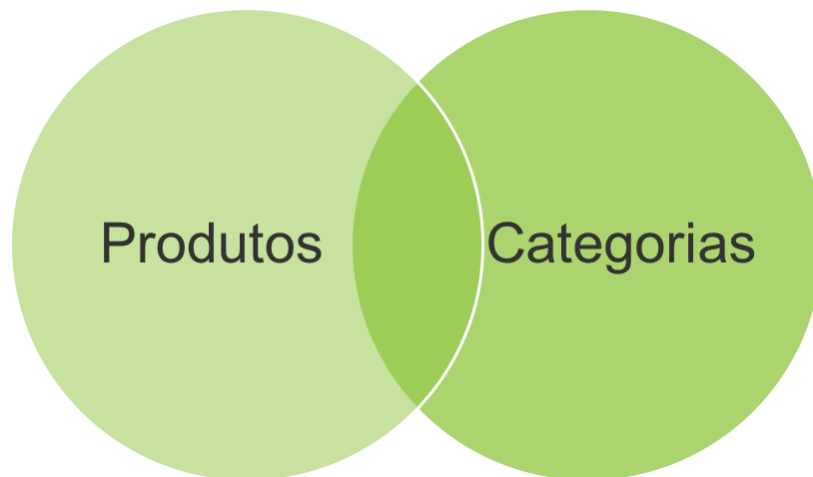
Nas pesquisas com **INNER JOIN** resultado trará somente as linhas que sejam comum nas 2 tabelas, ligadas pelos campos das tabelas em questão na pesquisa.

### LEFT JOIN



A cláusula **LEFT JOIN** permite obter não apenas os dados relacionados de duas tabelas, mas também os dados não relacionados encontrados na tabela à esquerda da cláusula JOIN. Caso não existam dados relacionados entre as tabelas à esquerda e a direita do JOIN, os valores resultantes de todas as colunas da lista de seleção da tabela à direita serão nulos.

## RIGHT JOIN



Ao contrário do **LEFT JOIN**, a cláusula **RIGHT JOIN** retorna todos os dados encontrados na tabela à direita do JOIN. Caso não existam dados associados entre as tabelas à esquerda e à direita de JOIN, serão retornados valores nulos.

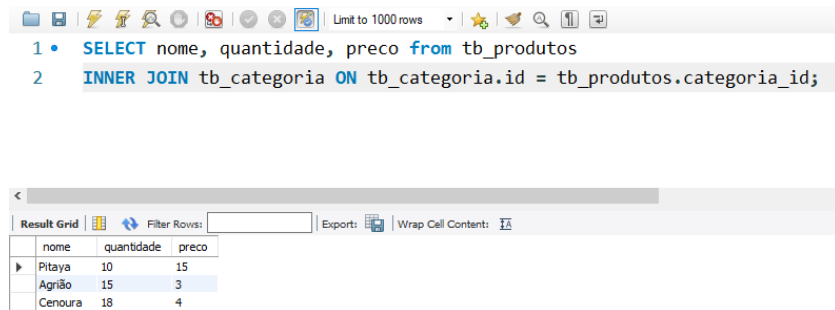
## Exemplos

Veremos os exemplos abaixo.

1. Digite a seguinte query para para **INNER JOIN**:

```
SELECT nome, quantidade, preco from tb_produtos  
INNER JOIN tb_categoria ON tb_categoria.id = tb_produtos.categoria_id;
```

E então você deverá ter um resultado como esse:



Limit to 1000 rows

```

1 • SELECT nome, quantidade, preco from tb_produtos
2 INNER JOIN tb_categoria ON tb_categoria.id = tb_produtos.categoria_id;

```

	nome	quantidade	preco
▶	Pitaya	10	15
	Agrião	15	3
	Cenoura	18	4

 [Documentação: Inner Join - W3Schools](#)

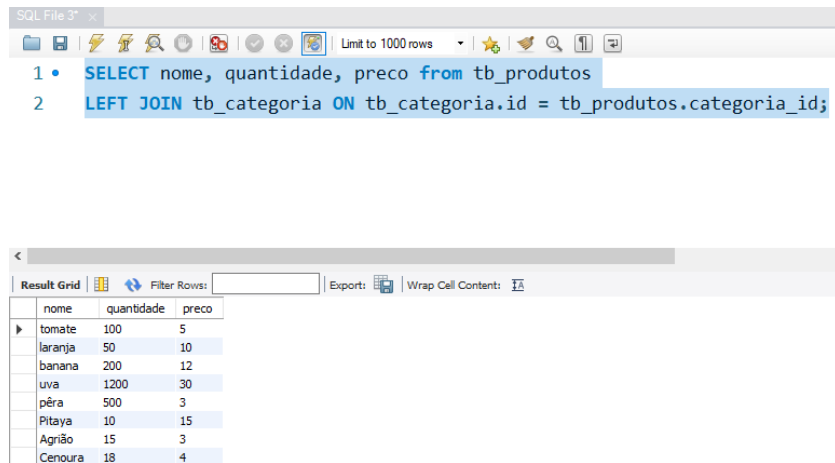
2. Digite a seguinte query para para **LEFT JOIN**:

```

SELECT nome, quantidade, preco from tb_produtos
LEFT JOIN tb_categoria ON tb_categoria.id = tb_produtos.categoria_id;

```

E então você deverá ter um resultado como esse:



SQL File 3\* x

Limit to 1000 rows

```

1 • SELECT nome, quantidade, preco from tb_produtos
2 LEFT JOIN tb_categoria ON tb_categoria.id = tb_produtos.categoria_id;

```

	nome	quantidade	preco
▶	tomate	100	5
	laranja	50	10
	banana	200	12
	uva	1200	30
	pêra	500	3
	Pitaya	10	15
	Agrião	15	3
	Cenoura	18	4

 [Documentação: Left Join - W3Schools](#)

3. Digite a seguinte query para para **RIGHT JOIN**:

```

SELECT descricao from tb_categoria
RIGHT JOIN tb_produtos ON tb_produtos.categoria_id = tb_categoria.id;

```

SQL File 3\* Queries do Cookie Book

Limit to 1000 rows

```
1 • SELECT descricao from tb_categoria
2 RIGHT JOIN tb_produtos ON tb_produtos.categoria_id = tb_categoria.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [↗](#)

	descricao
▶	NULL
	NULL
	NULL
	NULL
	NULL
	Fruta
	Verdura
	Legume



[Documentação: Right Join - W3Schools](#)