



# Manual de Querys no MySql

Esse manual tem fins educativos, foi realizado como forma de consulta para estudo

- 🔧 Criar Bd

- 🔧 Excluir Bd

- 🔧 Transferindo dados

- 🔧 Querys importantes:

- ★ Tabelas

- 🔧 Criando tabelas e colunas

- 🔧 Adicionar dados na tabela

- 🔧 Importantes:

- 🔧 Excluir Tabela

- 🔧 Renomear Tabela

- ★ Colunas

- 🔧 Adicionar uma nova coluna

- 🔧 Excluir coluna

- 🔧 Modificar o tipo e característica da coluna criada

- 🔧 Modificar nome da coluna

- ★ Linhas

- 🔧 Adicionar dados na linha ou modificar dados incorretos

- 🔧 Modificar mais linhas com os mesmos dados

- 🔧 Apagar linhas

- 🔧 Apagar TODAS as linhas de uma tabela

- 🔧 Gerenciando Cópias de Segurança:

- ★ Procedures

- 🔧 Criar Procedure

- 🔧 Executar a Procedure

- 🔧 Relacionamento de tabelas dentro da Procedure

- ★ SELECT

- 🔧 Ordenar de acordo com uma coluna

- 🔧 Ordenar descendente (de baixo para cima)

- 🔧 Ordenar ascendente (de cima para baixo)

- 🔧 Filtrar as colunas ordenando colunas

- 🔧 Filtrar Linhas

- 🔧 Usando operadores

- 🔧 Entre um e outro (BETWEEN, AND)

- 🔧 Where com mais dados (IN = em)

- 🔧 Combinando testes (AND, OR)

- 🔧 Puxar um dado

- 🔧 Puxar dois ou mais dados

- 🔧 Filtro de texto

- 🔧 Puxar dados sem repetir as informações

- 🔧 Função de agregação - Selecionar ou totalizar informações

- 🔧 Agrupar as informações

- 🔧 Agrupar informação especifica

- 🔧 Maior entre os dados

- 🔧 Menor entre os dados

- 🔧 Somar

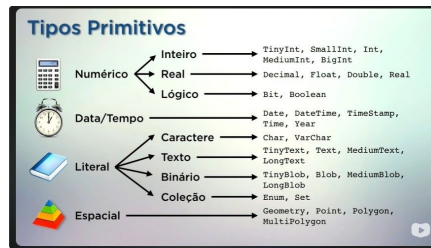
- 🔧 Media

Banco de dados é composto por tabelas > colunas > campos

Ao escolher os campos devemos definir seu tipo, chamamos de Tipos Primitivos. São basicamente os tipo de dados que iremos usar, por exemplo numéricos, data/hora, strings, espacial e etc.

[Tipos de dados MySql \(Link complementar\)](#)

[Tipos de dados MySql 2 \(Link complementar\)](#)



Tipos básicos mais utilizados

## 🔧 Criar Bd [↗](#)

```
1 create database nome_do_bd;

1 -- Cria um banco de dados --
2
3 • create database filhos;
```

## 🔧 Excluir Bd [↗](#)

```
1 drop database nome_do_bd;

6 -- Exclui um banco de dados --
7
8 • drop database filhos;
```

## 🔧 Transferindo dados [↗](#)

Não da para renomear um banco de dados, dessa forma criamos um bd novo e transferimos as tabelas e dados do antigo bd para a nova.

```
1 rename table nome_da_tabela to nome_do_bd_novo;

.

5 -- Transferir dados da tabela de um banco para outro --
6
7 • rename table cursos to filhos;
```

## 🚀 Querys importantes: [↗](#)

```
1 use nome_do_banco;
```

Abre o Banco

## 🌟 Tabelas [↗](#)

### 🔧 Criando tabelas e colunas [↗](#)

Ao criar a tabela já adicionamos as colunas com seus respectivos tipos

NOT NULL = significa que deverá ser preenchido, não aceita que fique nulo.

Auto\_increment = é um tipo onde se é auto incrementado de forma automática, nesse caso foi o Id

Campos com Chave Primária = A Primary Key é a chave que é utilizada como identificador único da tabela, sendo representada por aquele campo (ou campos), que não receberá valores repetidos. É como se fosse um CPF ou matrícula, um dado que não se repete.

```
7 -- Criando tabelas e colunas --
8 • create table dados (
9 id int NOT NULL AUTO_INCREMENT, -- campo com chave primaria --
10 nome varchar(30) not NULL, -- not null significa que deve ser preenchido esse dados, não aceita nulo --
11 nascimento DATE,
12 sexo enum('M','F'),
13 peso decimal(5,2),
14 PRIMARY KEY (id)
15 );
```

```
1 describe nome_da_tabela;
```

Apresenta a tabela criada com suas características e tipos

(podendo ser apenas **desc**)

Query

```
16 -- Descreve a tabela criada --
17 • describe dados;
```

Resultado

Result Grid

Filter Rows

Export

Wrap Cell Content

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI		auto_increment
nome	varchar(30)	NO			
nascimento	date	YES			
sexo	enum('M','F')	YES			
peso	decimal(5,2)	YES			

## 🔧 Adicionar dados na tabela 🔗

```
1 insert into nome_da_tabela
2 values (dados_novos);
```

-- Inserir dados na tabela --

- ```
• insert into dados
values
(default, 'Bruce', '2020-09-22', 'M', '6.8');
```

Nesse caso poderia também adicionar os campos que serão inseridos, exemplo:

```
1 insert into dados
2 (id, nome, nascimento, sexo, peso)
3 values
4 (default, 'Bruce', '2020-09-22', 'M', '6.8');
```

## 📌 Importantes: 🔗

**Default** = irá preencher sozinho

Posso colocar sem o id pois criei o comando de ir automático e sequencial, ou posso colocar default.

```
1 select *from nome_da_tabela;
```

Mostra os dados que foram adicionados de todas as colunas

Caso eu queira dados somente de uma coluna a query seria:

```
1 select nome, sexo from dados;
```

```
25 -- Mostra os dados que foi adicionado de todas as colunas --
26 • select *from dados;
27
```

| id | nome    | nascimento | sexo | peso |
|----|---------|------------|------|------|
| 2  | Bruce   | 2020-09-22 | M    | 6.80 |
| 3  | Castiel | 2020-09-22 | M    | 5.20 |
| 4  | Amora   | 2020-09-22 | F    | 5.90 |

## 🔧 Excluir Tabela 🔗

Só vai criar uma tabela nova no bd se ele não existir IF NOT EXISTS

Só vai apagar uma tabela ou um banco de dados se ele existir IF EXISTS

```
1 drop table if exists nome_da_tabela;
```

```
1 create table if not exists cursos (
2 nome varchar(30)
3 );
```

-- Apaga a tabela --

- ```
• drop table if exists filhos;
```

## 🔧 Renomear Tabela 🔗

```
1 alter table nome_da_tabela
2 rename to nome_da_nova_tabela;
```

```
1 alter table pessoa
2 rename to gafanhoto;
```

## Colunas [↗](#)

### Adicionar uma nova coluna [↗](#)

Adicionar a descrição e características dessa coluna

Depois AFTER

Primeiro FIRST

Caso seja a ultima coluna não é necessário adicionar nada

```
1 alter table nome_da_tabela
2 add column nome_da_coluna;
```

```
43 -- Adicionar uma nova coluna --
44
45 • alter table dados
46 add column cor varchar(10) after peso;
47
```

Result Grid | Filter Rows: | Edit:

	id	nome	nascimento	sexo	peso	cor
▶	2	Bruce	2020-09-22	M	6.80	NULL
	3	Castiel	2020-09-22	M	5.20	NULL
	4	Amora	2020-09-22	F	5.90	NULL
•	NULL	NULL	NULL	NULL	NULL	NULL

Nesse exemplo já foi add o tipo e onde ela irá ser adicionada, no caso foi após (after) da coluna peso, mas como seria a ultima coluna não seria necessário colocar o after poderia deixar sem nada.

### Excluir coluna [↗](#)

```
1 alter table nome_da_tabela
2 drop column nome_da_coluna;
```

### Modificar o tipo e característica da coluna criada [↗](#)

```
1 alter table nome_da_tabela
2 modify column nome_da_coluna;
```

```
1 alter table pessoa
2 modify column profissao varchar(20) not null default '';
```

### Modificar nome da coluna [↗](#)

```
1 ALTER TABLE nome_da_tabela
2 CHANGE column nome_atual_da_coluna novo_nome_da_coluna tipo_de_dados;
```

```
1 ALTER TABLE clientes
2 CHANGE idade idade_cliente INT;
```

## Linhas [↗](#)

### Adicionar dados na linha ou modificar dados incorretos [↗](#)

atualize = update

configure = set

onde = where

limit 1; = define quantas linhas poderão ser afetadas, limita a ação do meu comando

```
1 update nome_da_tabela
2 set nome_da_coluna
3 where nome_da_linha
```

Query

Resultado

```
50 • update dados
51 set cor = 'preto'
52 where id = '2';
53
```



--	--

Query	Resultado
<pre> 54 • update dados 55   set cor = 'branco' 56   where id = '4'; </pre>	

Query	Resultado
<pre> 58 • update dados 59   set cor = 'tricolor' 60   where id = '4'; </pre>	

Outros exemplos:

```

69 • update cursos
70   set descricao = 'Curso de HTML5'
71   where idcurso = '1';
72
73 • update cursos
74   set nome = 'PHP', ano = '2015'
75   where idcurso = '4';
76
77 • update cursos
78   set nome = 'Java', ano = '2015', carga = '40'
79   where idcurso = '5'
80   limit 1;

```

### 🛠️ Modificar mais linhas com os mesmos dados 🔗

```

1 update cursos
2   set ano = '2050', carga = '800'
3   where ano = '2018';

```

### 🛠️ Apagar linhas 🔗

```

1 delete from cursos
2   where idcurso= '8';
3
4 delete from cursos
5   where ano='2050'
6   limit 2;
7
8 delete from cursos
9   where idcurso= '10';

```

### 🛠️ Apagar TODAS as linhas de uma tabela 🔗

```

1 truncate nome_da_tabela;

```

### 🔗 Gerenciando Cópias de Segurança: 🔗

1. server - data export - seleciona o banco e as tabelas que quer exportar
2. dump structure and data (vai exportar o dado e estrutura)
3. export to self-contained file
4. include create schema

Irá guardar na pasta tudo o que usei a estrutura e os dados podendo levar de um servidor a outro.

Gera um dump completo e importa para outro

## 🔧 Procedures [↗](#)

Conjunto de comandos SQL que podem ser executados de uma só vez

Permitem ganhar tempo

```
--VERIFICANDO SE EXISTE PROCEDURE JÁ CRIADA NA BASE DE DADOS
--SE TIVER A PROCEDURE NA BASE TRÁ APAGAR PARA CRIAR NOVAMENTE
IF EXISTS (SELECT 1 FROM SYS.OBJECTS WHERE TYPE = "P" AND NAME = "SP_CONSULTAR_CLIENTES")
BEGIN
    DROP PROCEDURE SP_CONSULTAR_CLIENTES
END
GO

--STORED PROCEDURE SIMPLES SEM PARAMETROS
CREATE PROCEDURE SP_CONSULTAR_CLIENTES
AS
SELECT * FROM CLIENTES
GO
```

Nesse caso existia essa procedure, então realizou o drop dela

## 🔧 Criar Procedure [↗](#)

```
1 create procedure nome_doqueaprocedureiráfzer
2 AS
3 Select *from nome_da_tabela
4 go
```

## 🔧 Executar a Procedure [↗](#)

```
1 exec nome_daProcedure
```

## 🔧 Relacionamento de tabelas dentro da Procedure [↗](#)

```
--VERIFICANDO SE EXISTE PROCEDURE JÁ CRIADA NA BASE DE DADOS
--SE TIVER A PROCEDURE NA BASE TRÁ APAGAR PARA CRIAR NOVAMENTE
IF EXISTS (SELECT 1 FROM SYS.OBJECTS WHERE TYPE = "P" AND NAME = "SP_CONSULTAR_VENDAS_CLIENTES")
BEGIN
    DROP PROCEDURE SP_CONSULTAR_VENDAS_CLIENTES
END
GO

--STORED PROCEDURE SEM PARAMETROS E COM RELACIONAMENTO DE TABELAS
CREATE PROCEDURE SP_CONSULTAR_VENDAS_CLIENTES
AS
SELECT * FROM CLIENTES A(NOLOCK)
INNER JOIN VENDAS B(NOLOCK) ON (B.ID_CLIENTE=A.ID_CLIENTE)
GO
```

## 🔧 SELECT [↗](#)

O \* todas as colunas (campos)

Seleciona todas as colunas da tabela cursos

```
1 select *from nome_da_tabela;
```

```
1 select nome_da_coluna, nome_da_coluna from nome_da_tabela;
```

## 🔧 Ordenar de acordo com uma coluna [↗](#)

```
1 select *from nome_da_tabela
2 order by nome_da_coluna;
```

## 🔧 Ordenar descendente (de baixo para cima) [↗](#)

```
1 select *from nome_da_tabela
2 order by nome_da_coluna desc;
```

## 🔧 Ordenar ascendente (de cima para baixo) [↗](#)

Pode adicionar asc ou não colocar nada

```
1 select *from nome_da_tabela
2 order by nome_da_coluna asc;
```

## 🔧 Filtrar as colunas ordenando colunas [↗](#)

```
1 select nome_da_coluna from nome_da_tabela
2 order by nome_da_coluna;
```

```
1 select nome, cargo, ano from cursos
2 order by ano, nome;
```

```
1 select nome, carga, ano from cursos;
```

### Filtrar Linhas [↗](#)

Posso substituir \* pelo nome da coluna

```
1 select *from nome_da_tabela
2 where nome_da_coluna = dado_da_linha
3 order by nome_da_coluna;
```

```
1 select *from cursos
2 where ano = 2016
3 order by nome;
```

### Usando operadores [↗](#)

=  
  
<=  
  
>=  
  
<> ou !=

```
1 select *from nome_da_tabela
2 where nome_da_coluna <= 2015
3 order by nome_da_coluna;
```

```
1 select nome, descricao from cursos
2 where ano <= 2015
3 order by nome;
```

### Entre um e outro (BETWEEN, AND) [↗](#)

```
1 select nome_da_coluna from nome_da_tabela
2 where nome_da_coluna between 2014 and 2016;
```

```
1 select nome, ano from cursos
2 where ano between 2014 and 2015
3 order by ano desc, nome;
```

### Where com mais dados (IN = em) [↗](#)

```
1 select nome, ano, descricao from cursos
2 where ano in (2014,2016)
3 order by ano;
```

### Combinando testes (AND, OR) [↗](#)

```
1 select nome, descricao, carga, totaulas from cursos
2 where carga < 35 and totaulas < 30
3 order by carga;
```

```
1 select nome, descricao, carga, totaulas from cursos
2 where carga < 35 or totaulas < 30
3 order by carga;
```

### Puxar um dado [↗](#)

```
1 select nome_da_coluna from nome_da_tabela
2 where nome_da_coluna ='nome_do_dado';
```

```
1 select nome, profissao, sexo from gafanhotos
2 where profissao = 'programador';
```



```
1 select nome, profissao, sexo from gafanhotos
2 where profissao in ('programador');
```

### Puxar dois ou mais dados [↗](#)

```
1 select nome, profissao, sexo from gafanhotos
2 where profissao in ('Programador') and sexo = 'M';
```

```
1 select nome, profissao, sexo from gafanhotos
2 where profissao = 'Programador' and sexo = 'M';
```

### Filtro de texto [↗](#)

like 'a%' - com essa letra no começo da palavra  
like '%a%' - com essa letra em qqlr lugar  
like '%a' - com essa letra no final da palavra  
not like 'a%'  
not like '%a%'  
not like '%a'

```
1 select nome, nascimento, nacionalidade, sexo from gafanhotos
2 where nome like 'J%' and sexo = 'F' and nacionalidade = 'Brasil';
```

### Puxar dados sem repetir as informações [↗](#)

```
1 select distinct nome_da_coluna from nome_da_tabela;
```

```
1 select distinct nacionalidade from gafanhotos;
```

```
1 select distinct carga from cursos
2 order by carga;
```

### Função de agregação - Selecionar ou totalizar informações [↗](#)

count ( \* )  
Irá contar a quantidade da informação  
No lugar de \* pode adicionar o nome da coluna desejada

```
1 select count(*)from cursos;
```

```
1 select count(*)from cursos
2 where carga >40;
```

### Agrupar as informações [↗](#)

group by

```
1 select nome_da_coluna count(*) from nome_da_tabela
2 group by nome_da_coluna;
```

```
1 select carga, count(nome) from cursos
2 group by carga;
```

```
3
4 select totaulas, count(*) from cursos
5 group by totaulas;
6
7 select ano, count(*) from cursos
8 group by ano;
```

### Agrupar informação específica [↗](#)

having

```
1 select ano, count(*) from cursos
2 group by ano
3 having count(ano) >= 5
4 order by count(*) desc;
```

### **Maior entre os dados** [↗](#)

max()

```
1 select max(carga) from cursos;
```

### **Menor entre os dados** [↗](#)

min()

```
1 select nome, min(totaulas) from cursos
2 where ano = '2016';
```

### **Somar** [↗](#)

sum()

```
1 select sum(totaulas) from cursos where ano = '2016';
2
```

### **Media** [↗](#)

avg ()