

✈ Frete Grátis a partir de 150  Primeira troca justa (regras)

 Até 12x no cartão

 Tabela de medidas

## MYSQL

# APRENDA COMANDOS BÁSICOS DO MYSQL



Camiseta  
Michael  
Corleone  
Masculina

PROMO

Disponível em: P, M, G, GG,  
EXG

~~R\$64,90~~ • **R\$49,90**

[<https://www.todoespacoonline.com/w/2014/10/comandos-basicos-mysql/michael-corleone-masculina/>]

 Ver opções

[<https://www.todoespacoonline.com/w/2014/10/comandos-basicos-mysql/michael-corleone-masculina/>]



Camiseta  
Chaves  
Professor  
Linguica  
Girafales  
Masculina

PROMO

Disponível em: P, M, G, GG,  
EXG

~~R\$64,90~~ • **R\$49,90**

[<https://www.todoespacoonline.com/w/2014/10/comandos-basicos-mysql/chaves-professor-linguica-girafales-masculina/>]

 Ver opções

[<https://www.todoespacoonline.com/w/2014/10/comandos-basicos-mysql/chaves-professor-linguica-girafales-masculina/>]



Camiseta We  
Don't Believe in  
Humans  
Masculina

PROMO

Disponível em: P, M, G, GG,  
EXG

~~R\$64,90~~ • **R\$49,90**

[<https://www.todoespacoonline.com/w/2014/10/comandos-basicos-mysql/we-dont-believe-in-humans-masculina/>]

 Ver opções

[<https://www.todoespacoonline.com/w/2014/10/comandos-basicos-mysql/we-dont-believe-in-humans-masculina/>]

Neste artigo vou ser bem simples e direto, e tentarei explicar de maneira breve os comandos mencionados anteriormente. Além disso, vou explicar uma maneira bem simples para que você possa treinar tais comandos diretamente em um servidor MySQL com o phpMyAdmin.

## INSTALANDO O PHPMYADMIN

Para continuar seguindo o tutorial abaixo, vamos instalar o phpMyAdmin para que você tenha uma interface simples e visual dos comandos que você estiver executando na sua base de dados MySQL.

Na verdade, não há segredos na instalação do phpMyAdmin, já que qualquer servidor WAMP que você configurar em seu computador trará a instalação do mesmo. Como já criamos tutoriais sobre a instalação de um servidor WAMP e sobre a utilização do phpMyAdmin, recomendo que siga os tutoriais abaixo para concluir a operação:

Melhor sistema gestão de  
Rel. Gartner recém-lançar

Anúncio Só 5 soluções de gestão  
empresariais classificadas líder  
hyland.com

Learn more

- Crie tabelas e bases de dados no phpMyAdmin  
[<https://www.todoespacoonline.com/w/2014/07/bases-de-dados-no-phpmyadmin/>]
- EasyPHP: Aprenda a criar um servidor Wamp Local  
[<https://www.todoespacoonline.com/w/>]

Feito isso, você terá o phpMyAdmin instalado em seu computador para que possamos continuar.

Dentro do phpMyAdmin, você precisará acessar a opção “SQL” (no menu superior do lado direito).



A caixa de texto acima permite que você execute consultas MySQL como se estivesse fazendo por linha de comando.

## ENTENDENDO COMANDOS MYSQL

Apesar de não ser necessário colocar ponto e vírgula (;) na maioria dos programas gerenciadores MySQL (nem no PHP com PDO

[\[https://www.todoespacoonline.com/w/\]](https://www.todoespacoonline.com/w/)) é interessante que você nunca se esqueça de coloca-lo ao final de cada consulta que realizar manualmente, pois este é o modo correto. Portanto, todo e qualquer comando MySQL que você executar, deve terminar com um ponto e vírgula (;).

Exemplo:

```
CREATE DATABASE `minha_base_de_dados`;
```

Para os nomes de tabelas, bases de dados e campos, é interessante colocar dois acentos ao redor dos nomes, já que se o nome de sua tabela ou base de dados coincidir com qualquer comando MySQL, ocorrerá um erro e sua consulta não será realizada.

Exemplo:

Os nomes de comandos devem estar em letras maiúsculas.

Valores devem vir entre aspas. Exemplo:

```
SELECT `campo` FROM `tabela` WHERE `campo` = 'Valor';
```

Essas são as dicas básicas sobre os comandos MySQL.

## CRIANDO UMA BASE DE DADOS

Para criar uma base de dados utilizando comandos MySQL, utilizamos o seguinte:

```
CREATE DATABASE `nome_da_base_de_dados`;
```

Onde “*nome\_da\_base\_de\_dados*” seria exatamente o nome da base de dados que você deseja criar.

Pode ocorrer da base de dados que você estiver tentando criar já existir no seu servidor MySQL. Se isso ocorrer, você verá um erro no seguinte formato:

```
Can't create database 'nome_da_base_de_dados'; database exists
```

Que significa que a base de dados que você está tentando criar já existe.

Para resolver este erro, podemos fazer o seguinte:

```
CREATE DATABASE IF NOT EXISTS `nome_da_base_de_dados`;
```

O “IF NOT EXISTS” faz aquele erro se tornar um “*Warning*”, o que não afeta sua consulta em nada.

```
CREATE TABLE IF NOT EXISTS `nome_da_base_de_dados`.`nome_da_tabela` (  
  `campo_id` INT (11) NOT NULL AUTO_INCREMENT,  
  PRIMARY KEY (`campo_id`)  
);
```

Este é o comando mais básico que existe para criarmos uma tabela, já que você deve ter pelo menos um campo na mesma.

A parte inicial do comando é bem intuitiva, veja:

```
/*  
 * Crie a tabela nome_da_tabela na base de dados  
 * nome_da_base_de_dados se a mesma não existir  
 */  
CREATE TABLE IF NOT EXISTS `nome_da_base_de_dados`.`nome_da_tabela`
```

Logo em seguida, entre parênteses, adicionamos os campos e os detalhes dos campos.

```
/*  
 * Campo 'campo_id'  
 * INT (11) - Inteiro, Máximo de 11 casas  
 * NOT NULL - Campo não pode ser nulo (valor em branco)  
 * AUTO_INCREMENT - Campo é incrementado automaticamente  
 */  
`campo_id` INT (11) NOT NULL AUTO_INCREMENT,
```

Podemos adicionar quantos campos quisermos no mesmo formato, por exemplo:

```
CREATE TABLE `nome_da_base_de_dados`.`nome_da_tabela` (  
  `campo_id` INT (11) NOT NULL AUTO_INCREMENT,  
  `campo_texto_limitado` VARCHAR (255),
```

## Veja o mesmo comando comentado:

```
/*
 * Crie a tabela nome_da_tabela na base de dados
 * nome_da_base_de_dados se a mesma não existir
 */
CREATE TABLE IF NOT EXISTS `nome_da_base_de_dados`.`nome_da_tabela` (

/*
 * Campo 'campo_id'
 * INT (11) - Inteiro, Máximo de 11 casas
 * NOT NULL - Campo não pode ser nulo (valor em branco)
 * AUTO_INCREMENT - Campo é incrementado automaticamente
 */
`campo_id` INT (11) NOT NULL AUTO_INCREMENT,

/*
 * Campo 'campo_texto_limitado'
 * VARCHAR (255) - Campo de texto, máximo de 255 caracteres
 */
`campo_texto_limitado` VARCHAR (255),

/*
 * Campo 'campo_texto_ilimitado'
 * TEXT - Texto
 */
`campo_texto_ilimitado` TEXT,

/*
 * Campo 'campo_numerico'
 * INT (11) - Inteiro com 11 casas
 */
`campo_numerico` INT (11),
```

```
    * DEFAULT '0000-00-00 00:00:00' - Valor padrão 0000-00-00 00:00:00
    */
`campo_data` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',

/*
    * PRIMARY KEY (`campo_id`) - Chave primária 'campo_id'
    */
PRIMARY KEY (`campo_id`)

)

/*
    * Charset padrão UTF8
    */
CHARSET = utf8 ;
```

Os detalhes estão todos nos comentários.

## ALTERANDO A TABELA

Para alterar uma tabela já criada, utilizamos o seguinte:

```
ALTER TABLE `nome_da_base_de_dados`.`nome_da_tabela`
    ADD COLUMN `novo_campo` VARCHAR (255);
```

Melhor sistema gestão de  
Rel. Gartner recém-lançar

Anúncio Compare as principais  
gestão de documentos. O mais  
hyland.com

Learn more

```
ALTER TABLE `nome_da_base_de_dados`.`nome_da_tabela`  
DROP `novo_campo` ;
```

Poderá fazer isso com qualquer campos que preferir.

Caso queira alterar um campo que já existe:

```
ALTER TABLE `nome_da_base_de_dados`.`nome_da_tabela`  
CHANGE `campo_texto_limitado` `novo_nome` VARCHAR (10);
```

Agora o “campo\_texto\_limitado” se chamará “novo\_nome” e será limitado em 10 caracteres.

## APAGANDO A TABELA

Para apagar a tabela, faça o seguinte:

```
DROP TABLE `nome_da_base_de_dados`.`nome_da_tabela`;
```

E pronto, já era!

## INSERT – INSERINDO VALORES NA TABELA

Agora que você já aprendeu a manipular bases de dados e tabelas, crie novamente a tabela que você “dropou” (apagou) acima e vamos inserir valores nos campos.

O comando INSERT funciona da seguinte maneira:

```
INSERT INTO `nome_da_base_de_dados`.`nome_da_tabela` (  
    `campo_texto_limitado`,  
    `campo_texto_ilimitado`,  
    `campo_numerico`,
```



```
'Um texto qualquer! <p>Até HTML se preferir</p>',  
'50',  
'2014-10-21 12:15:41'  
);
```

Ou seja, você descreve os campos que quer preencher e, em VALUES, descreve os valores na mesma ordem em que adicionou os campos. Além disso, VALUES também pode inserir vários campos ao mesmo tempo, basta adicionar os valores entre parênteses e separá-los por vírgula.

Veja:

```
INSERT INTO `nome_da_base_de_dados`.`nome_da_tabela` (  
    `campo_texto_limitado`,  
    `campo_texto_ilimitado`,  
    `campo_numerico`,  
    `campo_data`  
)  
VALUES  
(  
    'Valor 1',  
    'Texto 1',  
    '1',  
    NOW()  
) ,  
(  
    'Valor 2',  
    'Texto 2',  
    '2',  
    NOW()  
) ,  
(  
    'Valor 3',  
    'Texto 3',  
    '3',  
    NOW()  
)
```

Agora eu selecionarei e limparei na base de dados com alguns comandos.

**Observação:** [NOW\(\)](http://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html#function_now) [\[http://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html#function\\_now\]](http://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html#function_now) é uma função MySQL para retornar a data e hora atual.

## SELECT – LENDO VALORES CADASTRADOS NA TABELA

SELECT é o comando MySQL que utilizamos para selecionar valores da base de dados. Ele pode se tornar um comando extremamente complexo.

Com ele você pode unir valores de várias tabelas, utilizar funções e alias para retornar valores de cálculos e funções MySQL e muito mais. Hoje vamos ficar no básico, já que poderíamos escrever milhares de artigos apenas para descrever a complexidade do comando SELECT.

Para não precisarmos ficar descrevendo o nome completo da base de dados, tabela e campos que queremos selecionar, podemos descrever para o MySQL qual base de dados estamos utilizando da seguinte maneira:

```
USE `nome_da_base_de_dados`;
```

Assim, ao invés de escrever `nome\_da\_base\_de\_dados`.`nome\_da\_tabela` toda vez que precisarmos, podemos descrever apenas `nome\_da\_tabela`, já que o MySQL já sabe em qual base de dados estamos trabalhando.

Para selecionar os valores de uma tabela, utilizamos o seguinte:

```
SELECT
    `campo_id`,
    `campo_texto_limitado`,
    `campo_texto_ilimitado`,
    `campo_numerico`,
    `campo_data`
```

Ou seja: "Selecione campo\_id, campo\_texto\_intimado, campo\_texto\_intimado, campo\_numerico, campo\_data da tabela nome\_da\_tabela".

Isso retornará os valores que cadastramos com o comando "INSERT".

Se você for selecionar todos os campos da tabela (como eu fiz), pode simplificar a consulta fazendo o seguinte:

```
SELECT
    *
FROM
    `nome_da_tabela` ;
```

O asterisco significa "todos".

## ORDER BY – ORDENAÇÃO

O comando SELECT pode ser ordenado utilizando o seguinte:

```
SELECT
    *
FROM
    `nome_da_tabela`
ORDER BY campo_id ASC ;
```

Ou seja: "Ordene pelo campo\_id de forma crescente".

Caso queira ordenar de forma decrescente, utilize DESC.

```
SELECT
    *
FROM
    `nome_da_tabela`
ORDER BY campo_id DESC ;
```

```
SELECT *  
FROM  
    `nome_da_tabela`  
ORDER BY `campo_id` ASC,  
    `campo_data` DESC,  
    `campo_numerico` ASC ;
```

Só que (conforme a consulta acima) se os campos não coincidirem com o que você precisa, sua consulta entrará em “contradição” (por assim dizer) e o MySQL vai demorar mais para realizar a tarefa. Além disso, podem ocorrer efeitos indesejados caso você utilize a ordenação de forma incorreta.

Existem mais formas para ordenar valores no MySQL, como ordem randômica por exemplo:

```
SELECT  
    *  
FROM  
    `nome_da_tabela`  
ORDER BY RAND() ;
```

Mas vamos ficar no básico por hoje.

## WHERE – SELECIONANDO VALORES ESPECÍFICOS

Com os comandos UPDATE, DELETE e SELECT, você pode selecionar um valor específico na base de dados, ou seja, apenas uma linha.

Para isso, utilizamos o comando “WHERE”, veja:

```
SELECT * FROM `nome_da_tabela` WHERE `campo_texto_ilimitado` = 'Texto 2';
```

```
SELECT * FROM `nome_da_tabela` WHERE `campo_texto_ilimitado` LIKE 'Text%';
```

Agora todos os valores do campos `campo_texto_ilimitado` que contenham a palavra “Text” no início serão encontrados.

Você também pode ser mais específico quanto aos campos e procurar mais valores.

```
SELECT
    *
FROM
    `nome_da_tabela`
WHERE `campo_texto_ilimitado` LIKE 'Text%'
    AND `campo_numerico` = 3
    OR `campo_numerico` = 2;
```

Agora estamos verificando se o `campo_texto_ilimitado` tem a palavra “Text” no início e se o `campo_numerico` tem valor 3 ou 2. AND e OR são utilizados para descrever “E” e “OU” respectivamente.

**Observação:** Valores numéricos não precisam de aspas.

## UPDATE – EDITANDO VALORES

Para editar valores cadastrados na sua base de dados, utilizamos o seguinte:

```
UPDATE
    `nome_da_tabela`
SET
    `campo_texto_ilimitado` = 'Novo valor'
WHERE campo_numerico = 3 ;
```

Tudo o que já foi descrito sobre `WHERE` se aplica aqui também.

Para atualizar vários campos ao mesmo tempo, faça o seguinte:

```
UPDATE
    `nome_da_tabela`
SET
    `campo_texto_ilimitado` = 'Novo valor',
    `campo_numerico` = 3,
    `campo_texto_limitado` = 'Outra coisa qualquer'
WHERE campo_numerico = 3 ;
```

Agora, além do `campo_texto_ilimitado`, também estou atualizando `campo_numerico` e `campo_texto_limitado`.

## DELETE – APAGANDO VALORES DA BASE DE DADOS

Este é o comando mais perigoso para valores de tabelas MySQL, pois não tem **NENHUMA CONFIRMAÇÃO**. Uma vez apagados, valores não podem ser recuperados.

Apesar disso, é um dos comandos mais simples para serem executados.

Para apagar qualquer valor de uma tabela MySQL, simplesmente faça o seguinte:

```
DELETE FROM `nome_da_tabela` WHERE `campo_numerico` = 3;
```

Pronto! Agora todos os valores onde o `campo_numerico` for igual a 3 serão apagados.

**Dica:** Para garantir que apenas uma linha seja apagada, utilize a chave primária (nosso caso, `campo_id`), já que é garantido que ela tenha o valor único daquela linha em toda a tabela.

## CONCLUINDO

Em caso de dúvidas, deixe um comentário.

---

21 DE OUTUBRO DE 2014

POR LUIZ OTÁVIO MIRANDA

---

**TAGS:** BÁSICOS, COMANDOS, MYSQL

## COMPARTILHAR ISSO

