

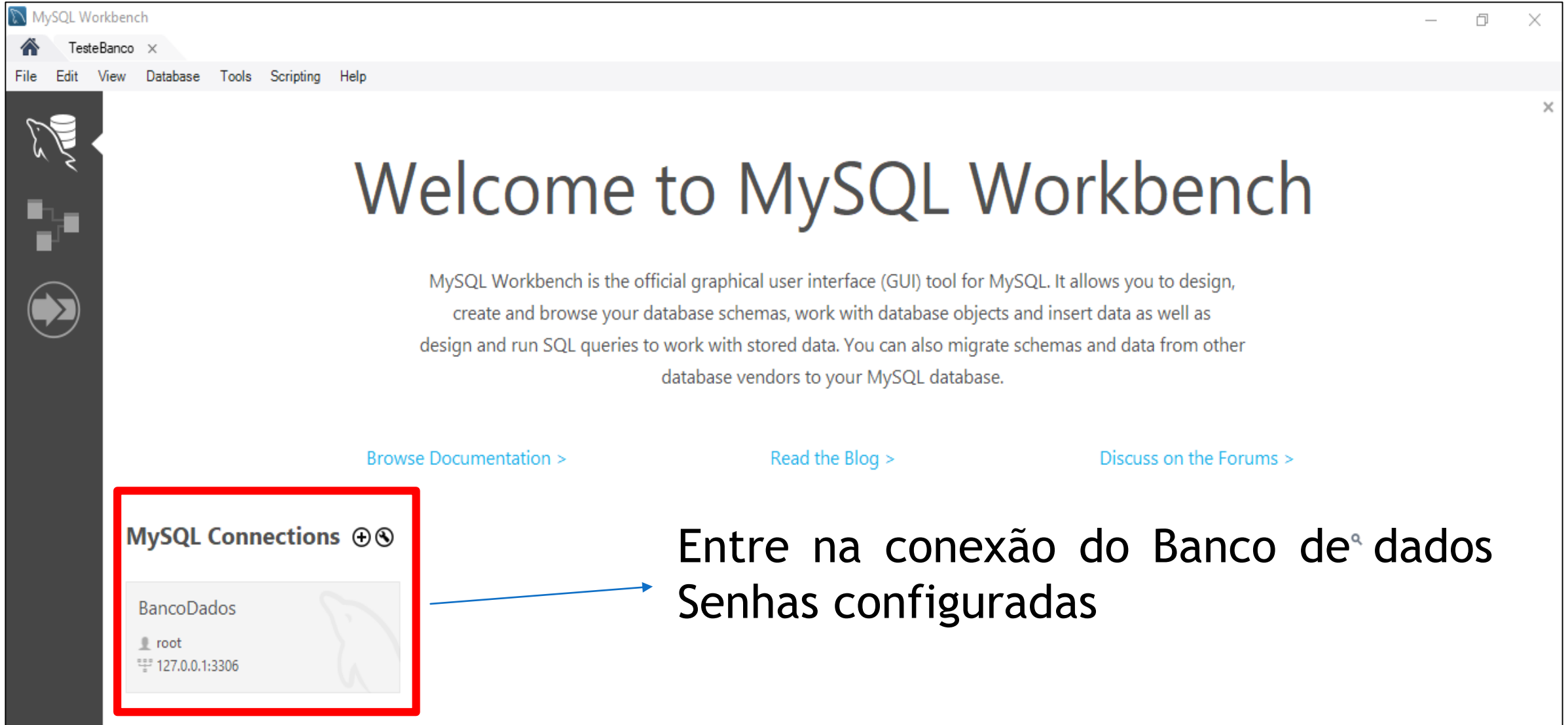


Comandos da linguagem SQL

Prof^a. Esp. Luciana Maria Oliveira

Recife, 2024

Conhecendo o Ambiente de Desenvolvimento



Comandos da Linguagem de consulta estruturada

O primeiro comando do SQL é a **criação de um banco de dados**. Mas, precisamos ficar atentos se já existe banco de dados com o nome que iremos criar, por esta razão devemos fazer uma verificação:

Para verificar se já existe, utilizaremos o comando:

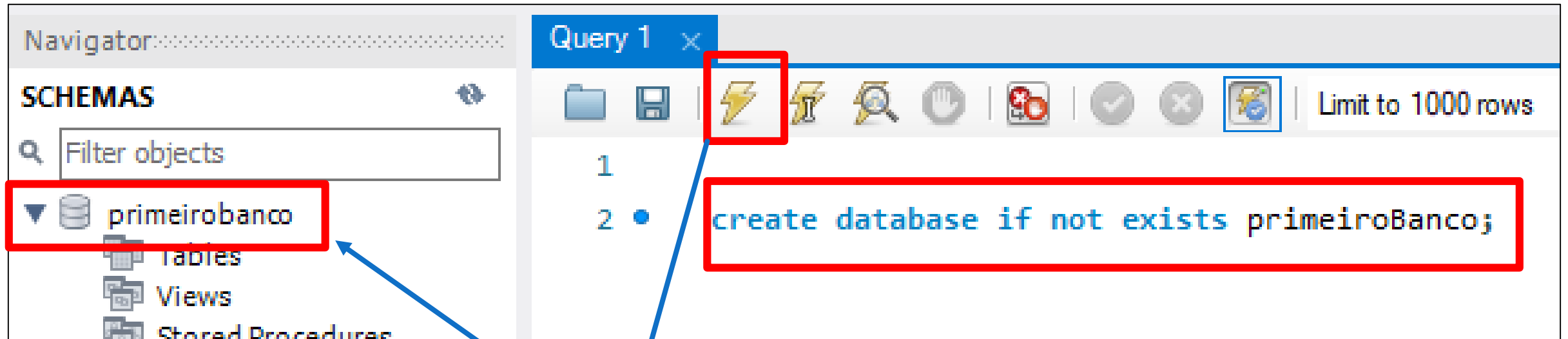
```
create database if not exists nome_banco;
```

Caso você tenha certeza que não existe um banco de dados, pode retirar a verificação:

```
create database nome_banco;
```

Comandos da Linguagem de consulta estruturada

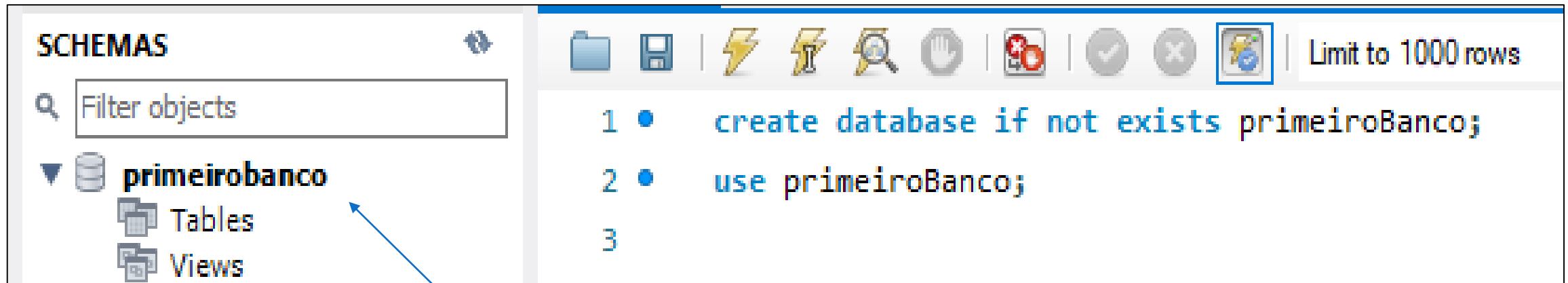
Exemplo do comando no ambiente de desenvolvimento



Após digitar os comandos, deverá clicar no raio para ativar a criação

Comandos da Linguagem de consulta estruturada

Para ativar o banco de dados deverá utilizar o comando: **use primeiroBanco;**



Quando o banco está em uso ele fica em negrito

Comandos da Linguagem de consulta estruturada

Para verificar quantos bancos de dados tem no ambiente de desenvolvimento:

Show databases;

4 • `show databases;`

<

Result Grid | Filter Rows:

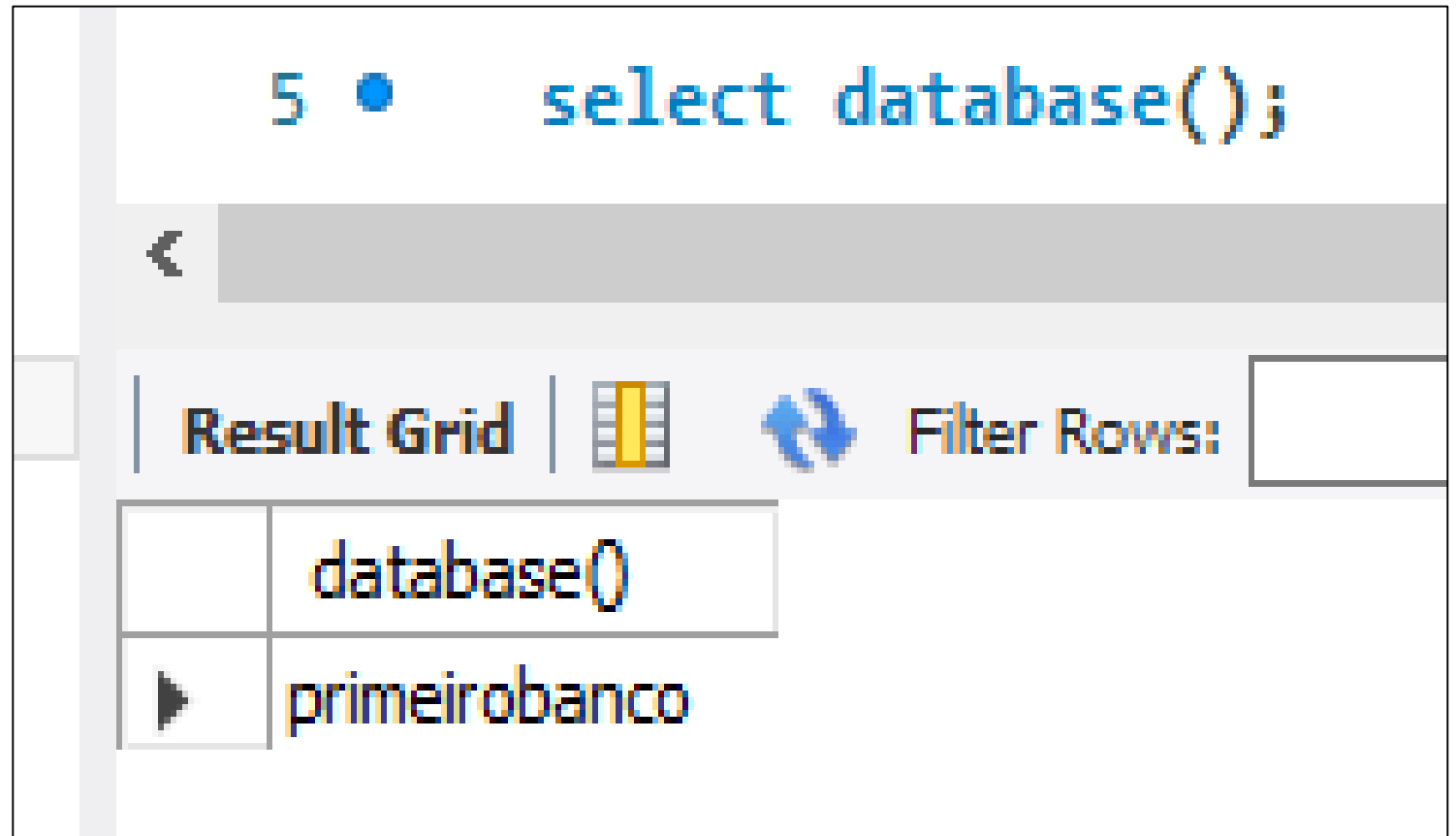
	Database
▶	information_schema
	mysql
	performance_schema
	primeirobanco
	sakila
	sys
	world

Banco criado anteriormente.

Comandos da Linguagem de consulta estruturada

Para mostrar qual banco de dados está sendo usado, use o comando:

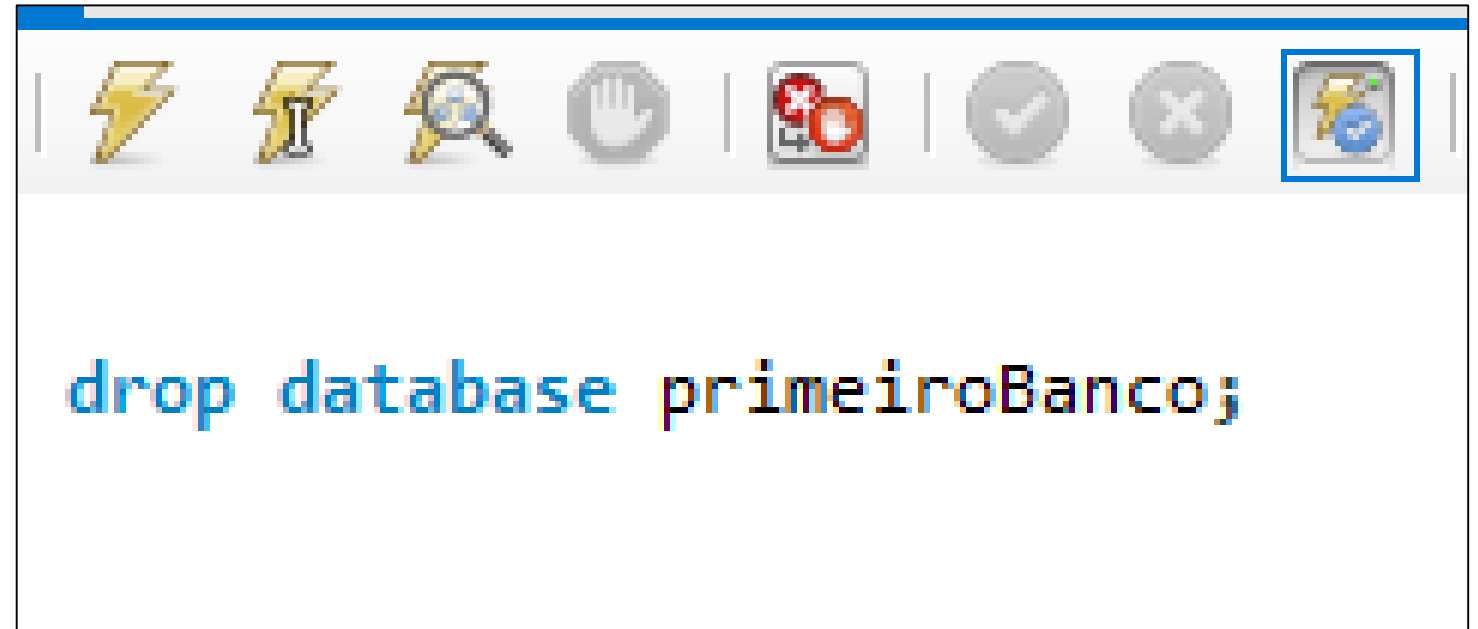
`select database();`



Comandos da Linguagem de consulta estruturada

Para remover um banco de dados use o comando:

drop database nomeBanco;





Criando uma tabela com chave primária ID

```
create database if not exists primeirobanco;  
use primeirobanco;
```

```
create table if not exists pessoas(  
  id int not null auto_increment,  
  nome varchar(30),  
  datanascimento date,  
  estado_civil enum('c', 's', 'd'),  
  peso decimal(5,2),  
  nacionalidade varchar(20) default 'Brasil',  
  primary key (id)  
);  
describe pessoas;
```

Result Grid


Filter Rows:

Export:


Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	nome	varchar(30)	YES		NULL	
	datanascimento	date	YES		NULL	
	estado_civil	enum('c','s','d')	YES		NULL	
	peso	decimal(5,2)	YES		NULL	
	nacionalidade	varchar(20)	YES		Brasil	

Lembrando que, quando você executar o comando pela segunda vez, ele irá mostrar uma mensagem de erro, identificando que já existe uma tabela criada. Deste modo, pode utilizar asterisco-barra : /* */ para comentar.

Inserindo dados na tabela

Para inserir um dado na tabela, utilizaremos o comando **insert into** nome_da_tabela **values()**; .

```
insert into pessoas(nome, datanascimento, estado_civil, peso, nacionalidade)
values
('Paulo', '1994-02-03', 's', '65.6', 'Paraguai')
```








```
select * from pessoas;
```

	id	nome	datanascimento	estado_civil	peso	nacionalidade
▶	1	Paulo	1994-02-03	s	65.60	Paraguai
⊙	NULL	NULL	NULL	NULL	NULL	NULL

Inserindo dados na tabela

Deixando de colocar o atributo nacionalidade que tem como default: Brasil

```
--  
20 • insert into pessoas(nome, datanascimento, estado_civil, peso)  
21 values  
22 ('Maria', '1994-02-03', 's', '55.6');  
23  
24 • select * from pessoas;  
25
```

<						
Result Grid  Filter Rows: <input type="text"/> Edit:    Export/Import:   Wrap Cell Content: 						
	id	nome	datanascimento	estado_civil	peso	nacionalidade
▶	1	Paulo	1994-02-03	s	65.60	Paraguaui
	2	Maria	1994-02-03	s	55.60	Brasil
•	NULL	NULL	NULL	NULL	NULL	NULL

Inserindo dados na tabela

Mas, também podemos colocar o nome DEFAULT para identificar que existe algo pré-definido, sem precisar remover o atributo.

```
insert into pessoas(id, nome, datanascimento, estado_civil, peso, nacionalidade)
values
(default, 'Carlos', '1995-02-03', 's', '79', default);

select * from pessoas;
```

nome	datanascimento	estado_civil	peso	nacionalidade
Paulo	1994-02-03	s	65.60	Paraguai
Maria	1994-02-03	s	55.60	Brasil
Maria	1994-02-03	s	55.60	Brasil
Carlos	1995-02-03	s	79.00	Brasil

Inserindo dados na tabela

Vale lembrar a cada execução o comando de inserção irá adicionar uma linha com os dados repetidos, desta forma, recomenda-se comentar após a primeira inserção.

Para comentar, usa-se o asterisco barra e barra asterisco.

`/* */`

```
/*
insert into pessoas2 (nome, datanascimento, idade, sexo, peso, nacionalidade)
values ('Paulo', '1994-02-03', '24', 'M', '78.5', 'Paraguai');

insert into pessoas2 (nome, datanascimento, idade, sexo, peso)
values ('João', '1994-02-03', '24', 'M', '78.5');

insert into pessoas2 (id, nome, datanascimento, idade, sexo, peso, nacionalidade)
values (default, 'Joana', '1992-02-03', '29', 'F', '78.5', default);
*/
```

Inserindo vários dados de um só vez

Para realizar a inserção de dados de uma só vez utiliza-se:

```
create table dados(  
    id int not null auto_increment,  
    nome varchar(30),  
    idade tinyint,  
    primary key(id)  
);
```

```
insert into dados (nome, idade)  
values  
('João', '24'),  
('maria', ' 34'),  
('carla', '22');
```

```
desc dados;
```

```
41 • select * from dados;
```

Result Grid




Filter Rows:

	id	nome	idade
▶	1	João	24
	2	maria	34
	3	carla	22
⊗	NULL	NULL	NULL

Inserindo nova coluna

Para adicionar um atributo da tabela pessoas2 deve seguir o seguinte comando:

```
16 • alter table pessoas2 add column profissao varchar(10);  
17  
18 • desc pessoas2;
```


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

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	nome	varchar(30)	NO		NULL	
	datanascimento	date	YES		NULL	
	idade	tinyint	YES		NULL	
	sexo	enum('F','M')	YES		NULL	
	peso	decimal(5,2)	YES		NULL	
	nacionalidade	varchar(20)	YES		Brasil	
	profissao	varchar(10)	YES		NULL	

Aviso que ocorreu a alteração sem problemas

Result 16 x

Output

 Action Output

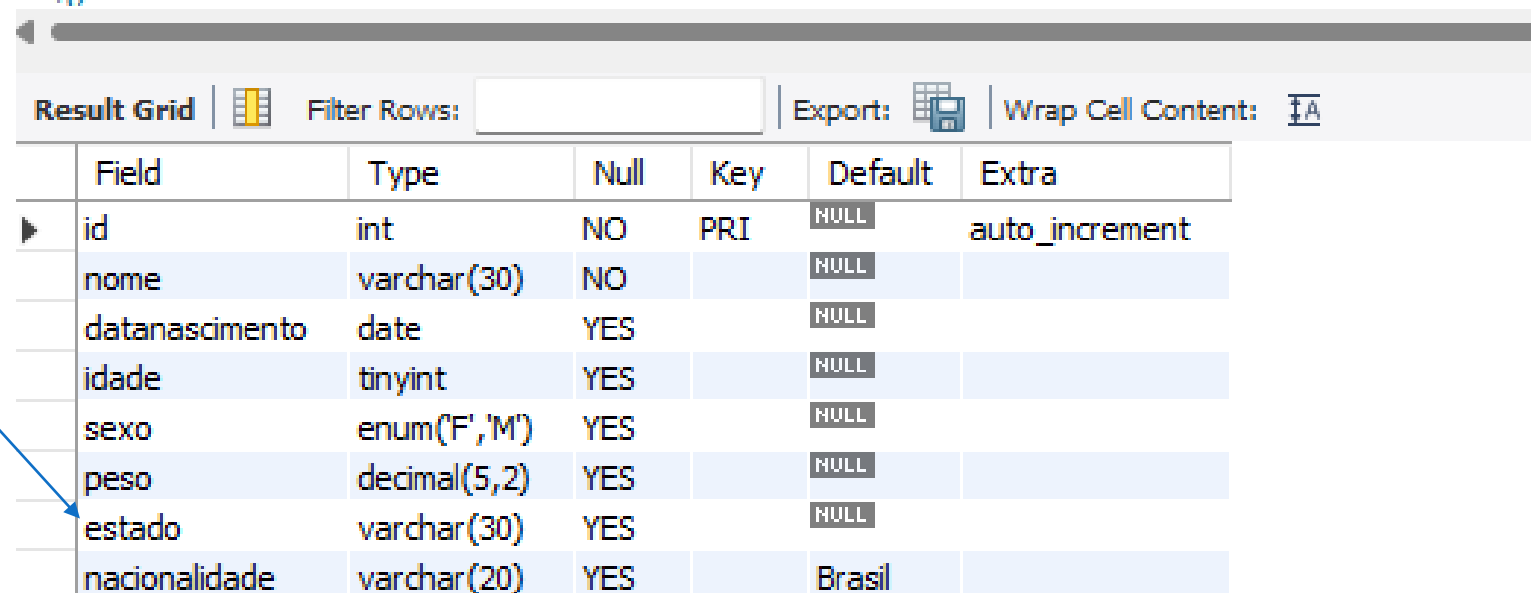
#	Time	Action
✓ 155	15:34:40	select * from pessoas2 LIMIT 0, 1000
✓ 156	15:34:40	desc pessoas2
⚠ 157	15:39:21	create database if not exists primeirobanco
✓ 158	15:39:21	use primeirobanco
✓ 159	15:39:21	alter table pessoas2 add column profissao varchar(10)
✓ 160	15:39:21	desc pessoas2

Alterando a posição do dado

```
21
22 • alter table pessoas2 add column estado varchar(30) after peso;
23
24 • desc pessoas2;
25
```

Adicionando uma coluna depois de peso

Mas, para isso o atributo não pode existir, senão dará erro de duplicação.







	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	nome	varchar(30)	NO		NULL	
	datanascimento	date	YES		NULL	
	idade	tinyint	YES		NULL	
	sexo	enum('F','M')	YES		NULL	
	peso	decimal(5,2)	YES		NULL	
	estado	varchar(30)	YES		NULL	
	nacionalidade	varchar(20)	YES		Brasil	

Criando mais uma tabela

```
28 • create table if not exists cursos(  
29     nome varchar(30) not null unique, —————→  
30     descricao varchar(20),  
31     carga int,  
32     totalaulas int,  
33     ano year default '2023'  
34 );
```

NOT NULL = não pode ser vazio

UNIQUE: Não pode conter dois nomes iguais.

<					
Result Grid					
Filter Rows: <input type="text"/>					
Edit:    Export/Import: 					
	nome	descricao	carga	totalaulas	ano
•	NULL	NULL	NULL	NULL	NULL

Criando chave primária após criar a tabela

```
35
36 • alter table cursos add column idcurso int first;
37 • alter table cursos add primary key(idcurso);
38
```

<

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	idcurso	int	NO	PRI	NULL	
	nome	varchar(30)	NO	UNI	NULL	
	descricao	varchar(20)	YES		NULL	
	carga	int	YES		NULL	
	totalaulas	int	YES		NULL	

Removendo colunas

Para a remoção deverá utilizar o comando DROP.

```
19
20 • alter table pessoas2 drop profissao;
21 • desc pessoas2;
22
```

Result Grid Filter Rows: Export: Wrap Cell Content:						
	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	nome	varchar(30)	NO		NULL	
	datanascimento	date	YES		NULL	
	idade	tinyint	YES		NULL	
	sexo	enum('F','M')	YES		NULL	
	peso	decimal(5,2)	YES		NULL	
	estado	varchar(30)	YES		NULL	
	nacionalidade	varchar(20)	YES		Brasil	

Modificando definições

23

24 • `alter table pessoas2 modify column estado varchar(10);`

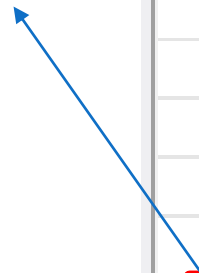
25 • `desc pessoas2;`

26

27

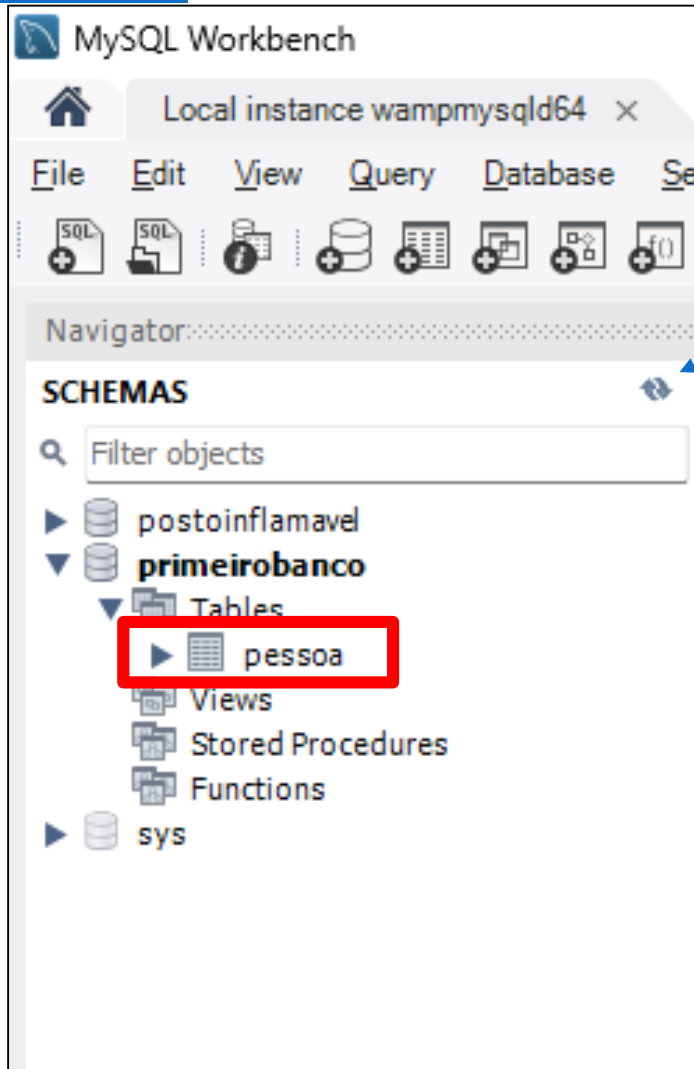
Tamanho
para 10

alterado



Result Grid Filter Rows: Export: Wrap Cell Content:						
	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	nome	varchar(30)	NO		NULL	
	datanascimento	date	YES		NULL	
	idade	tinyint	YES		NULL	
	sexo	enum('F','M')	YES		NULL	
	peso	decimal(5,2)	YES		NULL	
	estado	varchar(10)	YES		NULL	
	nacionalidade	varchar(20)	YES		Brasil	

Renomeando o nome da tabela



Atualiza o banco

```
alter table pessoas2 rename to pessoa;
```

Aplicando as Formas Normais



As Formas Normais são regras que são aplicadas dentro de um Banco de Dados para verificar se o Banco foi bem projetado. Também são chamadas de normalizações de Banco.

O Banco de dados é considerado normalizado se seguir as três regras de normalização.

Primeira Forma Normal (1FN)

Uma tabela estará seguindo a primeira forma normal (1FN) se todos os campos forem simples, ou seja, não possuir multivalorados (vários valores).

Exemplo: Tabela Cliente

Código	Nome	Telefone	Endereço
001	José	994884 887575	Rua nove, castelo branco. 234
002	Paulo	099442	Rua 12, praça da liberdade, 345

Não está
normalizada



Dois problemas: o telefone é multivalorado e o endereço é composto

Como resolver???

Primeira Forma Normal (1FN)

Resolver Endereço: Deverá criar coluna para cada campo.

Código	Nome	Telefone	Rua	Bairro	Número
001	José	994884 887575	Rua nove	castelo branco	234
002	Paulo	099442	Rua 12	praça da liberdade	345

Foi realizada a separação dos dados, para aplicar a primeira Forma Normal 1FN

Primeira Forma Normal (1FN)

Resolver Telefone: Criar outra tabela, já que nem todos os campos telefone irá ser preenchidos.

Código	Nome	Telefone
001	José	994884 887575
002	Paulo	099442

Primeiro cliente tem 2 telefones

Segundo cliente tem 1 telefone

Código	Telefone
001	994884
001	887575
002	099442

Aplicando a Segunda Forma Normal



Segunda Forma Normal (2FN)

Uma entidade estará na 2 Forma Normal se ela já ter passado pela 1FN e todos os seus atributos não chave **forem totalmente dependentes** da chave primária.

Tabela Pedidos

N_pedido	Codigo_pedido	Produto	quantidade	valor	subtotal
001	1-234	Computador	5	1.500	7500
002	2-789	impressora	3	350.0	1.050

→ Codigo_pedido e produto não é dependente da chave primária e valor é dependente de produto e não de pedido.

Segunda Forma Normal (2FN)

Aplicando a 2FN:

Tabela Pedidos

N_pedido	quantidade	subtotal
001	5	7500
002	3	1.050

Tabela Produto

Codigo_pedido	Produto	valor
1-234	Computador	1.500
2-789	impressora	350.0

Aplicando a Terceira Forma Normal

Terceira Forma Normal (3FN)

Terceira forma normal 3FN é remover os campos que pode ser obtidos pela equação de outros campos da tabela.

Tabela Pedidos

N_pedido	quantidade	subtotal
001	5	7500
002	3	1.050

Tabela Produto

Codigo_pedido	Produto	valor
1-234	Computador	1.500
2-789	impressora	350.0



Não é necessário este campo de **subtotal**, pois se multiplicarmos o campo **quantidade** (tabela pedidos) com o **valor** (tabela produto), teríamos o resultado sem a necessidade de ter um campo específico para armazenar este resultado.

Terceira Forma Normal (3FN)

Aplicando a Terceira forma normal 3FN:

Tabela Pedidos

N_pedido	quantidade
001	5
002	3

Manipulando Linhas (UPDATE - DELETE E TRUNCATE)

Registros- Linhas

- `create database if not exists banco_dados;`
- `use banco_dados;`
- `create table if not exists cursos_ofertados(
id int not null auto_increment,
nome varchar(30),
quantidade_vagas int,
primary key (id)
);`

	id	nome	quantidade_vagas
▶	1	Informática	5
	2	matemática	3
	3	lógica	10
	4	Designer	6
✱	NULL	NULL	NULL

- `insert into cursos_ofertados(nome, quantidade_vagas)
values
('Informática', '5'),
('matemática', '3'),
('lógica', '10'),
('Designer', '6');`
- `select * from cursos_ofertados;`

UPDATE

Atualizar a quantidade de vagas em lógica, colocar 8

	id	nome	quantidade_vagas
▶	1	Informática	5
	2	matemática	3
	3	lógica	10
	4	Designer	6
✱	NULL	NULL	NULL

```
update cursos_ofertados set quantidade_vagas = '8' where id = '3';  
select * from cursos_ofertados;
```

nome	quantidade_vagas
Informática	8
matemática	3
lógica	8

Modificando vários registros através do Update







	id	nome	quantidade_vagas
▶	11	lógica	10
	12	Designer	6
	13	Informática	5
	14	matemática	3
	15	lógica	10
	16	Designer	6
	17	Informática	5
	18	matemática	5
	19	lógica	10
	20	Designer	5

Modificar os 3 registros que tenha a quantidade de vagas igual a 5.

Modificando vários registros através do Update

```
26 • update cursos_ofertados set quantidade_vagas = '10' where quantidade_vagas = '5' limit 3;  
27 • select * from cursos_ofertados;
```

28

<			
Result Grid			
Filter Rows: <input type="text"/>			
Edit:   			
Export/Import:  			
Wrap Cell Content: 			
	id	nome	quantidade_vagas
	13	Informática	10
	14	matemática	3
	15	lógica	10
	16	Designer	6
	17	Informática	10
	18	matemática	10
	19	lógica	10
	20	Designer	10
	21	Informática	5

Excluindo registros indesejados

id	nome	quantidade_vagas
1	Informática	8
2	matemática	3
3	lógica	8
4	Designer	6
5	Informática	10
6	matemática	3
7	lógica	10
8	Designer	6
9	Informática	10

Delete da tabela cursos_ofertados a linha 2 matemática contendo uma quantidade de vagas de 3.

```
delete from cursos_ofertados where id = '2';  
select * from cursos_ofertados;
```

Result Grid			
	id	nome	quantidade_vagas
▶	1	Informática	8
	3	lógica	8
	4	Designer	6
	5	Informática	10

Excluindo registros indesejados

Excluir várias linhas que tenha quantidade igual a 8.

```
31 • delete from cursos_ofertados where quantidade_vagas = '8' limit 3;  
32 • select * from cursos_ofertados;  
33
```

Depois

Result Grid			
Filter Rows: <input type="text"/>			
	id	nome	quantidade_vagas
▶	4	Designer	6
	5	Informática	10
	6	matemática	3
	7	lógica	10
	8	Designer	6

Antes

	id	nome	quantidade_vagas
	1	Informática	8
	3	lógica	8
	4	Designer	6
	5	Informática	10


Truncando uma tabela

Truncar uma tabela é apagar todos os registros de uma tabela.

```
32 • truncate cursos_ofertados;
33 • select * from cursos_ofertados;
```

34

<

Result Grid |  Filter Rows: | Edit

	id	nome	quantidade_vagas
•	NULL	NULL	NULL

Após apagado não tem mais volta, cuidado ao utilizar esse comando.

Comando SELECT

Comando SELECT

Selecione o nome dos cursos por ordem alfabética:

```
select * from cursos_ofertados order by nome;
```

Tabela Antes

	id	nome	quantidade_vagas
▶	1	Informática	5
	2	matemática	3
	3	lógica	10
	4	Designer	6
●	NULL	NULL	NULL

Tabela Depois

	id	nome	quantidade_vagas
▶	4	Designer	6
	8	Designer	6
	1	Informática	5
	5	Informática	5
	3	lógica	10
	7	lógica	10

Comando SELECT

Selecione apenas o registro nome da tabela

```
select nome from cursos_ofertados;
```

	id	nome	quantidade_vagas
▶	4	Designer	6
	8	Designer	6
	1	Informática	5
	5	Informática	5
	3	lógica	10
	7	lógica	10



	nome
▶	Informática
	matemática
	lógica
	Designer
	Informática
	matemática
	lógica
	Designer

Conhecendo o comando LIKE

O comando **LIKE**, significa: semelhança, igualdade, parecido.

Exemplo: Selecione apenas os nomes que começa com a letra minúscula 'm%'.

	id	nome	quantidade_vagas
▶	1	Informática	5
	2	matemática	3
	3	lógica	10
	4	Designer	6
	5	Informática	5
	6	matemática	3
	7	lógica	10
	8	Designer	6
	9	medicina	2
	10	massagem	4
●	NULL	NULL	NULL

Select * from cursos_ofertados where nome like 'm%'.

	id	nome	quantidade_vagas
▶	2	matemática	3
	6	matemática	3
	9	medicina	2
	10	massagem	4
●	NULL	NULL	NULL

A porcentagem significa carta coringa, o primeiro será o m e a porcentagem completará com o restante do nome, não esqueça de colocar entre aspas simples.

Conhecendo o comando count

Saber a quantidade de cursos cadastrados.

```
select count(*) from cursos_ofertados;
```

```
select count(quantidade_vagas) from cursos_ofertados;
```

	id	nome	quantidade_vagas
▶	1	Informática	5
	2	matemática	3
	3	lógica	10
	4	Designer	6
	5	Informática	5
	6	matemática	3
	7	lógica	10
	8	Designer	6
	9	medicina	2
	10	mensagem	4
.	NULL	NULL	NULL



	count(*)
▶	10



	count(quantidade_vagas)
▶	10

Conhecendo o comando max, min

Saber o mínimo e o máximo de quantidade de vagas

```
select max(quantidade_vagas) from cursos_ofertados;  
select min(quantidade_vagas) from cursos_ofertados;
```

	id	nome	quantidade_vagas
▶	1	Informática	5
	2	matemática	3
	3	lógica	10
	4	Designer	6
	5	Informática	5
	6	matemática	3
	7	lógica	10
	8	Designer	6
	9	medicina	2
	10	massagem	4
.	NULL	NULL	NULL



	max(quantidade_vagas)
▶	10



	min(quantidade_vagas)
▶	2

Agora é sua vez!

- Crie um banco de dados com um nome de sua preferência;
- Deverá criar uma tabela aluno;
- Os atributos de aluno são: id do aluno, nome, sobrenome, idade e cargo. O ID será inteiro, não nulo e auto incrementável. O cargo terá como padrão: 'Não definido', o nome será varchar(30) e os demais realize a sua maneira.
- Deverá realizar a inserção dos dados de 10 alunos. Utilize o comando insert. Utilize a formatação DEFAULT em 3 alunos;
- Após isso, adicione um novo atributo a tabela aluno, o atributo será: estado, a posição dele é depois de idade.
- Altere o tamanho do nome para varchar(10).
- Crie uma outra tabela chamada de cursos sem chave primária que contenha 3 colunas (nome, quantidade, turno);
- Agora utilize o comando de alter table para adicionar a chave primária nessa tabela;
- Preencha essa tabela com 5 dados;
- Selecione dados dessa tabela ordenado pelo nome.
- Crie um relacionamento (chave estrangeira) dessa tabela com a tabela aluno, para fazer isso primeiro você tem que adicionar uma coluna na tabela aluno, com o nome referência a chave estrangeira. Por exemplo deverá criar na tabela aluno um atributo chamado id_curso int.
- Agora na tabela aluno crie um relacionamento de chave estrangeira.
- Pesquise como você irá retornar as informações de nome do aluno e nome do curso.
- Exclua um registro da tabela aluno;