```
# para comentários em uma só linha, usamos #
/* para comentários com mais de uma linha, usamos /* no começo e, no final, usamos */

/*
este é um exemplo de comentário com mais de uma linha.
Veja que, enquanto eu não fecho, continua sendo comentário.
Agora, vou fechar com */

/*
Dicas que vou usar nestes exemplos:
	para rodar um comando, aperte CTRL + Enter (dica do João Pedro!)
	para rodar um comando SQL somente, mesmo com tantos outros escritos abaixo, basta selecionar o comando;
	se for usar o botão de raio acima, o primeiro roda todos os comandos
		mas isso pode dar ruim, caso algum comando já tenha sido executado antes
	o segundo botão de raio, que tem um número junto, roda apenas o comando onde está o cursor
*/

/*==================================Primeiro Exercício (fixação)================================*/

# vamos criar um novo banco (database) chamado aula_2
CREATE DATABASE aula_2;

#lembre-se sempre de usar o "ponto e vírgula", pessoal!

#agora, precisamos "usar" esse banco (para criar as tabelas nele e etc.):

# USE aula_2;

/*
vamos criar uma tabela chamada Pessoa, com identificador único (chave primária), nome e idade
dica: pulando linha e apertando "TAB", podemos "identar" o código, deixando tudo mais organizado
	isso faz com que o tudo fique mais fácil para entender em outros momentos
*/

/*
create table Pessoa (
	id_pessoa int primary key auto_increment,
   nome varchar(200),
   idade int
   );
 */

#então, vamos verificar como foi criada a tabela
```

```
# select * from Pessoa;

#vamos inserir na tabela uma pessoa chamada Creusa, com 20 anos de idade

# insert into Pessoa (nome, idade) values ('Creusa', 20);

#vamos supor que surgiu a necessidade de adicionar um atributo telefone na tabela
#recorremos ao comando "ALTER TABLE", para alterar a tabela

# alter table Pessoa add column telefone varchar(11);
#para conferir:
#select * from Pessoa;

#precisamos atualizar a Creusa inserindo o dado de telefone:
# update Pessoa set telefone = '68991111111' where id_pessoa = 1;
#para conferir:
# select * from Pessoa;

/*
para salvar o script feito até o momento, basta apertar Ctrl + S
vamos salvar como Aula_2 na área de trabalho
*/

/*
vamos inserir só mais duas pessoas:
Valéria, com 14 anos e telefone 68992222222
Dedé, com 75 anos e telefone 68993333333
*/
/*
INSERT INTO Pessoa (nome, idade, telefone)
	VALUES ('Valéria', 14, '68992222222'), ('Dedé', 75, '68993333333');
select * from Pessoa;
*/

/*==================================Segundo Exercício (um passo a
mais)================================*/

/*
vamos criar uma nova tabela 'funcionarios', com identificador único, nome, salário,
departamento...
desta vez, vamos deixar um tudo pouco mais detalhado:
*/

/*
create table funcionarios (
	id_funcionario int not null auto_increment, #=> 1 - ver detalhes abaixo sobre "not
null"
```

```
        nome varchar(200) not null,
    salario decimal(6,2) not null default '0', #=> 2 - ver detalhes abaixo sobre "decimal(6,2)" e
sobre "not null '0'"
    departamento varchar(50) not null,
    PRIMARY KEY (id_funcionario) #=> 3 - ver detalhes abaixo sobre primary key no final da
criação da tabela
    );
*/
/*
=> soobre os detalhes:
detalhe 1 - not null: esta coluna (atributo) não pode ter valor nulo
detalhe 2 - decimal(6,2): como vimos anteriormente, DECIMAL exige indicar casas antes de
depois da vírgula
detalhe 2 - default: é para inserção de um valor padrão
        para o caso de não haver um valor nesse atributo na hora de inserir o registro
            ele vai começar indicando 0 de salário
detalhe 3 - você pode definir a chave primária tanto no próprio atributo, quanto no final
*/

#vamos ver as tabelas que temos no banco aula_2
# show tables;

/*
repare que a tabela 'funcionarios' está em letra minúscula... será que podemos modificar
isso?
a resposta é sim.. podemos usando o comando RENAME
*/

/*
RENAME TABLE funcionarios to Funcionarios;
show tables;
*/

/*
vamos criar uma tabela Veiculos, com identificador único, marca, modelo, placa e
funcionário que vai usar o veículo
*/

/*
create table Veiculos (
        id_veiculo int not null auto_increment,
    marca varchar(50) not null default '', #=> 1 - detalhes abaixo sobre aspas simples vazias
após default
    modelo varchar(50) not null default '',
    placa varchar(7) not null default '',
    funcionario int default null, #=> 2 - detalhes abaixo sobre o atributo "funcionario"
    primary key (id_veiculo),
```

```
   CONSTRAINT fk_veiculos_funcionarios foreign key(funcionario) REFERENCES
Funcionarios(id_funcionario) #=> 3 - detalhes abaixo sobre tora linha do código neste
momento
   );
*/

/*
detalhe 1 - neste caso, o DEFAULT com aspas simples sem nada entre elas indica que o
campo aparecerá vazio caso não seja preenchido com a placa ao registrar
        detalhe: no insert, você não deverá indicar que vai inserir um "valor" para placa:

        INSERT INTO Veiculos () VALUES (); #=> veja que não fizemos "INSERT INTO
Veiculos (placa) VALUES ();"
                se tivéssemos determinado "placa" entre parênteses, teríamos que,
obrigatoriamente, ter um valor para inserir no campo "placa"

ao rodar o comando acima, você verá que há um registro na tabela Veiculos, mas com
todas as colunas vazias (default '')
*/
#vamos conferir:
# select * from Veiculos;

/*
detalhe 2 - o atributo 'funcionario' será a chave estrangeira da tabela Veiculos,
                permitindo o relacionamento da tabela Veiculos com a tabela Funcionarios
                observe que esse atributo (funcionario), na tabela Veiculos precisa ser INT
                        tem que ser igual como foi feito com o atributo id_funcionario, a chave
primária da tabela Funcionarios
        por padrão, permite valor nulo

detalhe 3 - vamos detalhar aqui:
                CONSTRAINT fk_veiculos_funcionarios foreign key(funcionario)
REFERENCES Funcionarios(id_funcionario)
                        CONSTRAINT
                                regra que você aplica a uma coluna para limitar o tipo de
dados que pode ser colocado nela
                                garante a precisão e a integridade dos dados
            em casos de identificar a chave estrangeira, CONSTRAINT garante a integridade
referencial
                        fk_veiculos_funcionarios
                                nome que você está dando à sua restrição de chave
estrangeira
            dar nomes descritivos às constraints é uma boa prática:
                                        para facilitar a identificação
                para facilitar o gerenciamento no futuro
                        FOREIGN KEY
                                palavra-chave que especifica que a coluna (funcionario) será
uma chave estrangeira
```

```
                        REFERENCES
                                Indica a tabela e a coluna que a chave estrangeira está
referenciando
*/

#Show tables;

/*
agora, você sabia que o Workbench pode criar um diagrama de tudo que criamos agora?
este é um mecanismo de 'engenharia reversa'
você seleciona na aba lá em cima "Database" e "Reverse Engineer..."
*/
```