



Unidade 20 – MySQL User Defined Function



Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com

Bibliografia

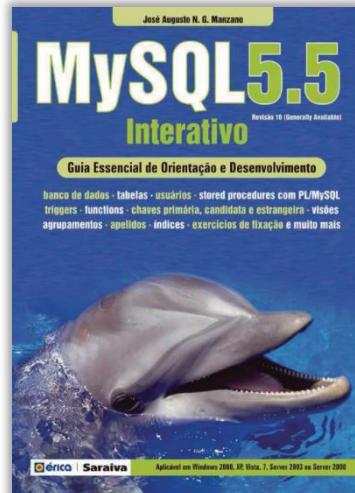
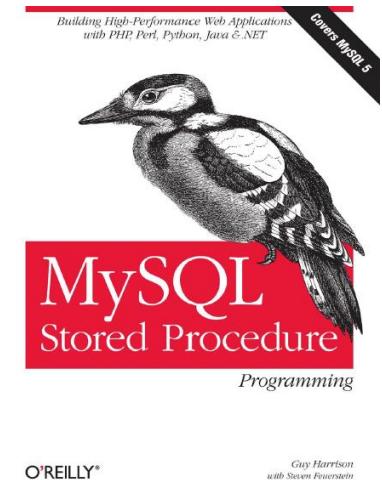
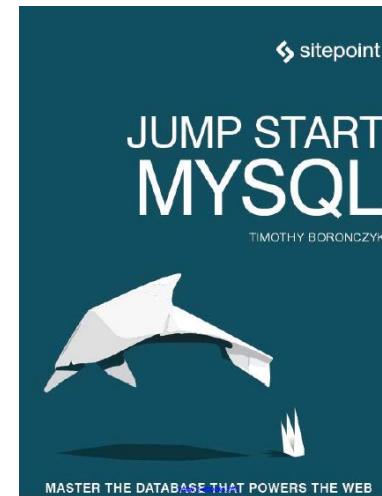
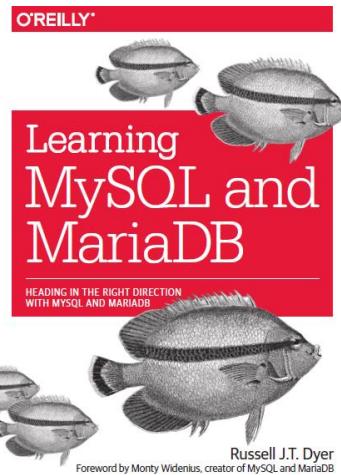
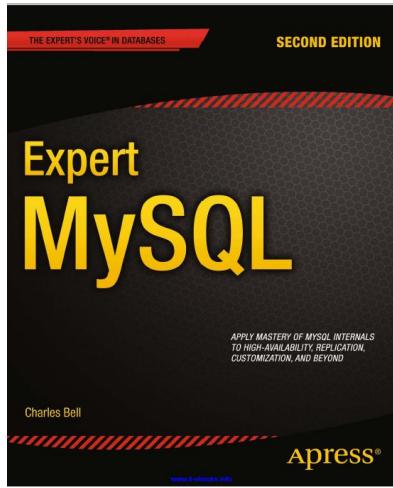
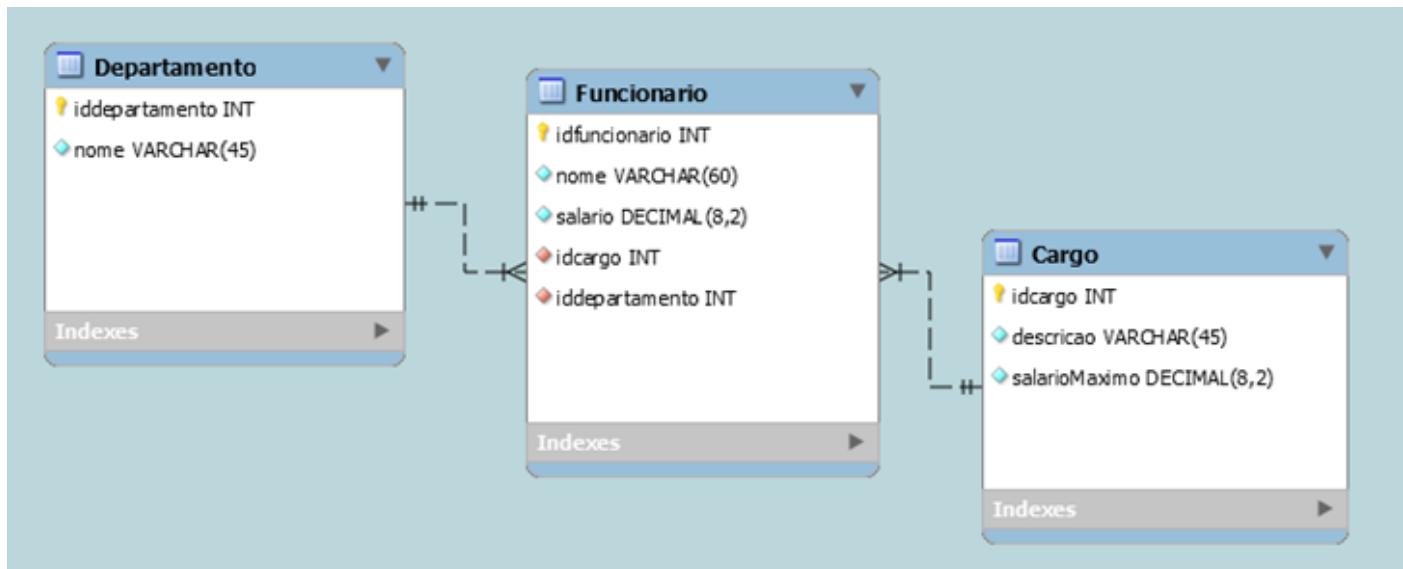


Diagrama Entidade Relacionamento

Este é o diagrama entidade relacionamento utilizado para essa unidade!



User Defined Function

```
CREATE DATABASE funcionario;

USE funcionario;

CREATE TABLE Cargo (
    idcargo INT(11) NOT NULL AUTO_INCREMENT,
    descricao VARCHAR(45) NOT NULL,
    salarioMaximo DECIMAL(8,2) NOT NULL,
    PRIMARY KEY (idcargo));

CREATE TABLE Departamento (
    iddepartamento INT(11) NOT NULL AUTO_INCREMENT,
    nome VARCHAR(45) NOT NULL,
    PRIMARY KEY (iddepartamento));

CREATE TABLE Funcionario (
    idfuncionario INT(11) NOT NULL AUTO_INCREMENT,
    nome VARCHAR(60) NOT NULL,
    salario DECIMAL(8,2) NOT NULL,
    idcargo INT(11) NOT NULL,
    iddepartamento INT(11) NOT NULL,
    PRIMARY KEY (idfuncionario),
    INDEX fk_Funcionario_Cargo_idx (idcargo ASC),
    INDEX fk_Funcionario_Departamento1_idx (iddepartamento ASC),
    CONSTRAINT fk_Funcionario_Cargo FOREIGN KEY (idcargo)
        REFERENCES Cargo (idcargo),
    CONSTRAINT fk_Funcionario_Departamento1 FOREIGN KEY (iddepartamento)
        REFERENCES Departamento (iddepartamento));
```



FuncionarioCriacaoDB.sql

Funções Embutidas

São funções **internas** ao **MYSQL** que se pode utilizar para operações matemáticas, datas, strings, etc.

<https://dev.mysql.com/doc/refman/5.7/en/functions.html>



[MOD \(\)](#) also works on values that have a fractional part and returns the exact remainder after division:

```
1 | mysql> SELECT MOD(34.5,3);
2 |       -> 1.5
```



Funções Embutidas

<https://dev.mysql.com/doc/refman/5.7/en/functions.html>



```
1 mysql> SELECT MINUTE('2008-02-03 10:05:03');
2      -> 5
```

```
1 mysql> SELECT MONTH('2008-02-03');
2      -> 2
```

```
1 mysql> SELECT MONTHNAME('2008-02-03');
2      -> 'February'
```

```
1 mysql> SELECT NOW();
2      -> '2007-12-15 23:50:26'
```

Funções Embutidas

Chapter 12 Functions and Operators

Table of Contents

- [12.1 Function and Operator Reference](#)
- [12.2 Type Conversion in Expression Evaluation](#)
- [12.3 Operators](#)
- [12.4 Control Flow Functions](#)
- [12.5 String Functions](#)
- [12.6 Numeric Functions and Operators](#)
- [12.7 Date and Time Functions](#)
- [12.8 What Calendar Is Used By MySQL?](#)
- [12.9 Full-Text Search Functions](#)
- [12.10 Cast Functions and Operators](#)
- [12.11 XML Functions](#)
- [12.12 Bit Functions and Operators](#)
- [12.13 Encryption and Compression Functions](#)
- [12.14 Information Functions](#)
- [12.15 Spatial Analysis Functions](#)
- [12.16 JSON Functions](#)
- [12.17 Functions Used with Global Transaction IDs](#)
- [12.18 MySQL Enterprise Encryption Functions](#)
- [12.19 Aggregate \(GROUP BY\) Functions](#)
- [12.20 Miscellaneous Functions](#)
- [12.21 Precision Math](#)



Funções String

- ✓ Executam operações em tipos de dados string, tais como: **VARCHAR**, **CHAR** e **TEXT**;

ASCII

→ **string1 = ASCII(string2)**

- ✓ A função **ASCII** retorna o código **ASCII** correspondente ao primeiro caractere do **string** passado por parâmetro;
- ✓ O exemplo a seguir, ilustra um uso da função **ASCII** em conjunto com as funções **LENGTH** e **CONCAT**;



Função ASCII

```
Function_01.sql •
F: > MySQL > MySQL_Functions > Hands_On_Functions > Function_01.sql

1
2 delimiter $$

3
4 create function function_01
5 (
6     in_string varchar(80)
7 )
8
9 )
10
11 returns varchar(256)
12
13 deterministic
14
15 begin
16
17     declare i int default 1;
18     declare string_len int;
19
20     declare out_string varchar(256) default '';
21
22     set string_len = LENGTH(in_string);
23
24     while (i <= string_len) do
25
26         set out_string = CONCAT(out_string, ASCII(substr(in_string, i, 1)), ' ');
27         set i = i + 1;
28
29     end while;
30
31     return (out_string);
32
33 end$$
34
35 delimiter ;
```

Função ASCII

```

Function_01.sql •
F:\MySQL>MySQL_Functions > Hands_On_Functions > Function_01.sql
1
2 delimiter $$ 
3
4 create function function_01
5 (
6     in_string varchar(80)
7 )
8
9 )
10
11 returns varchar(256)
12
13 deterministic
14
15 begin
16
17 declare i int default 1;
18 declare string_len int;
19
20 declare out_string varchar(256) default '';
21
22 set string_len = LENGTH(in_string);
23
24 while (i <= string_len) do
25
26     set out_string = CONCAT(out_string, ASCII(substr(in_string, i, 1)), ' ');
27     set i = i + 1;
28
29 end while;
30
31 return (out_string);
32
33 end$$
34
35 delimiter ;
36

```

```

mysql> set @x = function_01('abc');
Query OK, 0 rows affected (0.00 sec)

```

```

mysql> select @x;
+-----+
| @x   |
+-----+
| 97 98 99 |
+-----+
1 row in set (0.00 sec)

```

```

mysql>

```

Função CHAR

- ✓ Retorna os caracteres correspondentes a um ou mais códigos ASCII passados como parâmetro;

CHAR

➔ string = **CHAR(ascii code [, . . .])**

Função CHAR

```
Proc20_01.sql ×  
F: > MySQL > MySQL_Functions > Hands_On_Functions > Proc20_01.sql  
1  
2  
3 delimiter @@  
4  
5 create procedure proc20_01()  
6  
7 begin  
8  
9     declare i int default 1;  
10  
11    drop table if exists tabelaCaracteres;  
12  
13    create table tabelaCaracteres  
14        (  
15            codigo_ascii int ,  
16            caractere_ascii char(1)  
17        );  
18  
19    while ( i < 128 ) do  
20  
21        insert into tabelaCaracteres values ( i, char(i) );  
22  
23        set i = i + 1;  
24  
25    end while;  
26  
27 end@@  
28  
29 delimiter ;  
30
```

Função CHAR

```

Proc20_01.sql ×

F: > MySQL > MySQL_Functions > Hands_On_Functions > Proc20_01.sql

1
2
3 delimiter @@
4
5 create procedure proc20_01()
6
7 begin
8
9     declare i int default 1;
10
11    drop table if exists tabelaCaracteres;
12
13    create table tabelaCaracteres
14    (
15        codigo_ascii int ,
16        caractere_ascii char(1)
17    );
18
19    while ( i < 128 ) do
20
21        insert into tabelaCaracteres values ( i, char(i) );
22
23        set i = i + 1;
24
25    end while;
26
27 end@@
28
29 delimiter ;
30

```

```

mysql>
mysql> call proc20_01();
Query OK, 1 row affected (0.37 sec)

mysql> select * from tabelacaracteres;
+-----+-----+
| codigo_ascii | caractere_ascii |
+-----+-----+
|           1 | ☺
|           2 | ☻
|           3 | ♥
|           4 | ♦
|           5 | ♣
|           6 | ♠
+-----+-----+

```

Built-in Functions

- ✓ String
- ✓ Numericas
- ✓ Date e Time
- ✓ Generalistas



User Defined Function

- ✓ É um tipo especial de **stored procedure** que **retorna** um único **valor**;
- ✓ Enquanto **stored procedures** podem retornar valores por meio de variáveis **OUT** ou **INOUT**, uma **function** pode – e deve – retornar dados somente por um único **RETURN**;
- ✓ Pode-se utilizar as funções de encapsular fórmulas comuns ou de regras de negócios que possam ser **reutilizáveis** em instruções **SQL**;
- ✓ **Functions** podem melhorar a legibilidade e manutenabilidade dos programas, por meio do encapsulamento de regras de negócio;
- ✓ Pode-se também utilizá-las para controle do **fluxo** de execução do programa.

User Defined Function

```
CREATE FUNCTION fn_DobraSoma(varValorA INT, varValorB INT) RETURNS INT
```

- ✓ O processo para definição de uma **User Defined Function** é similar ao de uma **Stored Procedure**;
- ✓ Primeiro, é necessário definir o nome da função e em sequência declarar os parâmetros da função;
- ✓ Por padrão , todos os parâmetros são implicitamente parâmetros **IN**;
- ✓ Sua principal diferença com relação às **Stored Procedures** é que se deve especificar o **tipo** do valor de retorno na declaração, sendo possível utilizar qualquer tipo de dados **MySQL** válido.

User Defined Function

- ✓ Toda função deve conter uma instrução **RETURN**, que é responsável por retornar um valor para quem a invocou;
- ✓ Sempre que a declaração **RETURN** for processada, a execução da função armazenada é encerrada.



User Defined Function

- ✓ A maioria das opções para **CREATE FUNCTION** também se aplicam para **CREATE PROCEDURE**;
- ✓ Entretanto, para funções a cláusula **RETURNS** é mandatória e define o tipo de dado que a função irá retornar;
- ✓ Todos os parâmetros de uma **function** são implicitamente **IN**;
- ✓ O corpo de uma **function** deve conter um ou mais comandos **RETURN**, os quais terminam a **function** e retornam o resultado para o programa chamador.

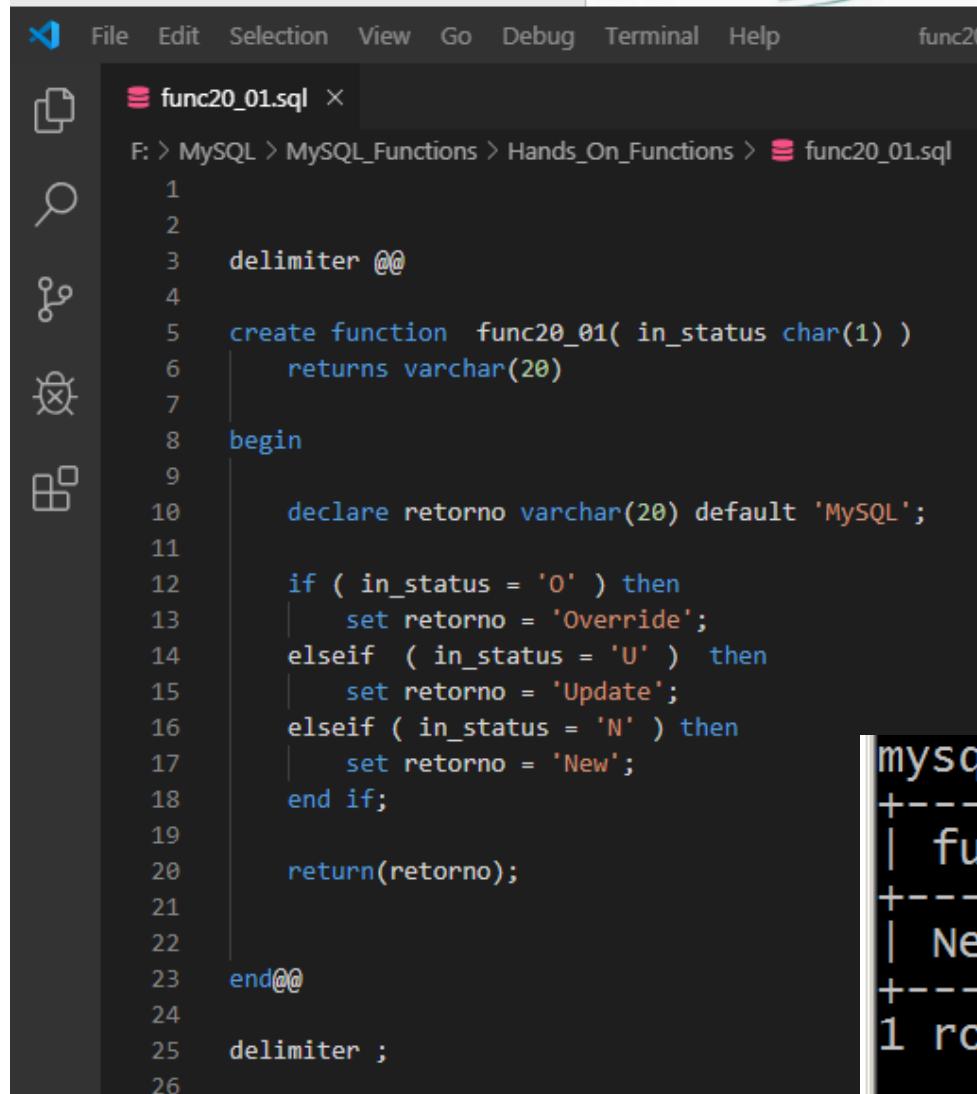
User Defined Function

```
CREATE FUNCTION fn_Dobro(varValor INT) RETURNS INT  
RETURN varValor + varValor;
```

```
SELECT fn_Dobro(4);
```

Result Grid	
	fn_Dobro(4)
▶	8

User Defined Function



The screenshot shows a software interface for writing MySQL code. The menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The title bar says "func20_01". The left sidebar has icons for file operations like Open, Save, Find, and Refresh. The main area contains the following SQL code:

```
1
2
3 delimiter @@
4
5 create function func20_01( in_status char(1) )
6     returns varchar(20)
7
8 begin
9
10    declare retorno varchar(20) default 'MySQL';
11
12    if ( in_status = '0' ) then
13        set retorno = 'Override';
14    elseif ( in_status = 'U' ) then
15        set retorno = 'Update';
16    elseif ( in_status = 'N' ) then
17        set retorno = 'New';
18    end if;
19
20    return(retorno);
21
22 end@@
23
24 delimiter ;
```



```
mysql> select func20_01('n');
+-----+
| func20_01('n') |
+-----+
| New           |
+-----+
1 row in set (0.00 sec)

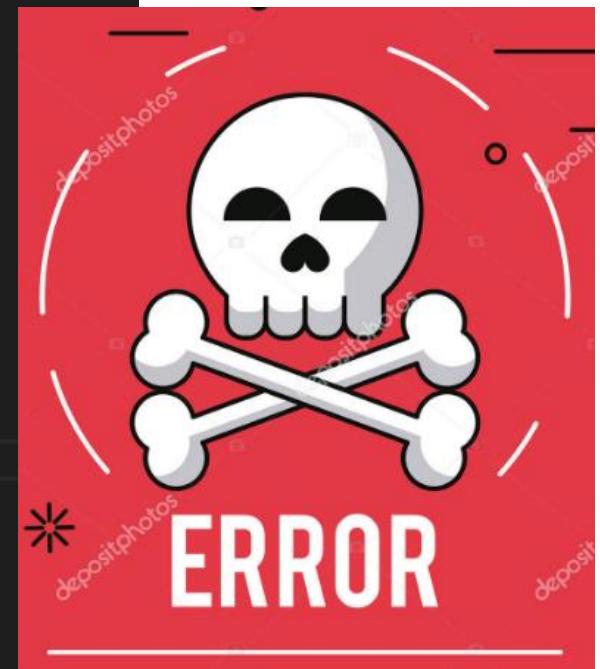
mysql>
```

User Defined Function

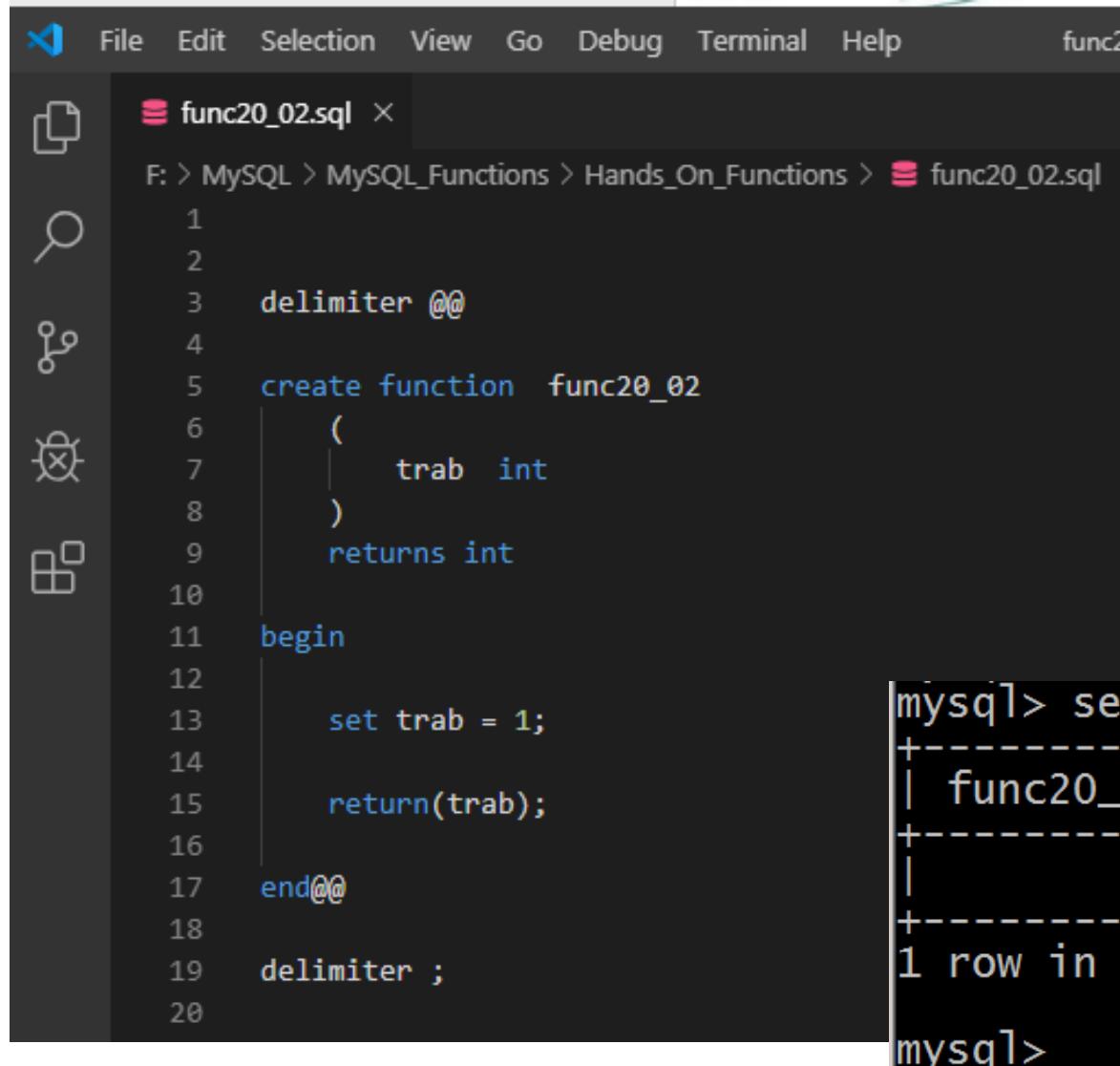


```
func20_02.sql
F: > MySQL > MySQL_Functions > Hands_On_Functions > func20_02.sql

1
2
3  delimiter @@
4
5  create function  func20_02
6  (
7      out trab  int
8  )
9      returns int
10
11 begin
12
13     set trab = 1;
14
15     return(trab);
16
17 end@@
18
19 delimiter ;
```



User Defined Function



```
func20_02.sql
```

```
F: > MySQL > MySQL_Functions > Hands_On_Functions > func20_02.sql
```

```
1
2
3  delimiter @@
4
5  create function  func20_02
6  (
7      |    trab  int
8      )
9      returns int
10
11 begin
12
13     set trab = 1;
14
15     return(trab);
16
17 end@@
18
19 delimiter ;
```

```
mysql> select func20_02(56);
+-----+
| func20_02(56) |
+-----+
|          1   |
+-----+
1 row in set (0.00 sec)

mysql>
```

User Defined Function

Sintaxe para exclusão de uma **User Defined Function**:

```
DROP FUNCTION fn_Dobro;
```

User Defined Function

Alguns exemplos para utilização de Funções:

- ✓ Select;
- ✓ Critérios de seleção;
- ✓ Avaliação de agregações;
- ✓ Atribuição de valores através da instrução **UPDATE**;
- ✓ Chamada através de **Stored Procedures**;
- ✓ Chamada através de **Funções**.

User Defined Function

Utilização de uma **Function** em um Select:

```
SELECT c.idCargo, c.Descricao, c.salarioMaximo, fn_Dobro(c.salarioMaximo)
FROM cargo c;
```

	idCargo	Descricao	salarioMaximo	fn_Dobro(c.salarioMaximo)
▶	1	Coordenador de TI	10954.00	21908
	2	Administrador de Rede	7087.00	14174
	3	Analista de Sistemas	6882.00	13764
	4	Analista de Segurança de Sistemas	5827.00	11654
	5	Coordenador de Suporte Técnico	8981.00	17962
	6	Analista de Suporte Técnico	4564.00	9128
	7	Técnico de Informática	2438.00	4876
	8	Estagiário (Penúltimo Ano)	1479.00	2958
	9	Estagiário (Último Ano)	1636.00	3272

User Defined Function

Utilização de uma **Function** em um critério de seleção:

```
SELECT c.idCargo, c.Descricao, c.salarioMaximo  
FROM cargo c  
WHERE c.idCargo = fn_Dobro(2);
```

	idCargo	Descricao	salarioMaximo
▶	4	Analista de Segurança de Sistemas	5827.00

User Defined Function

Utilização de uma **Function** em um critério de avaliação de agregações:

```
SELECT f.idCargo, c.descricao, COUNT(*) 'QtdeCargos'  
FROM funcionario f  
INNER JOIN cargo c ON c.idcargo = f.idcargo  
GROUP BY f.idCargo  
HAVING QtdeCargos >= fn_Dobro(1);
```

	idCargo	descricao	QtdeCargos
	8	Estagiário (Penúltimo Ano)	5
▶	9	Estagiário (Último Ano)	3

User Defined Function

Utilização de uma **Function** em uma instrução **UPDATE**:

```
UPDATE funcionario  
SET salario = fn_Dobro(5420)  
WHERE idfuncionario = 2;
```

User Defined Function

Utilização de uma **Function** em uma **Stored Procedure**:

```
DELIMITER $$  
CREATE PROCEDURE sp_DobraSalario(IN varIdFuncionario INT)  
BEGIN  
    UPDATE funcionario  
    SET salario = fn_Dobro(salario)  
    WHERE idfuncionario = varIdFuncionario;  
END$$  
DELIMITER ;
```

User Defined Function

Modularização

```
DELIMITER $$  
CREATE FUNCTION sp_DobraDobro(varQuant INT) RETURNS INT  
BEGIN  
    RETURN fn_Dobro(varQuant) + fn_Dobro(varQuant);  
END$$  
DELIMITER ;
```



	sp_DobraDobro(2)
▶	8

User Defined Function

Assim como ocorre em uma **Stored Procedure** em uma **User Defined Function** é possível a utilização de múltiplos parâmetros:

```
CREATE FUNCTION fn_DobraSomaTipos(varValorA INT,  
                                   varValorB DECIMAL(6,2)) RETURNS DECIMAL(6,2)  
RETURN (varValorA + varValorB) * 2;
```

```
SELECT fn_DobraSomaTipos(2, 3.54);
```

Result Grid	
	fn_DobraSomaTipos(2, 3.54)
▶	11.08

User Defined Function

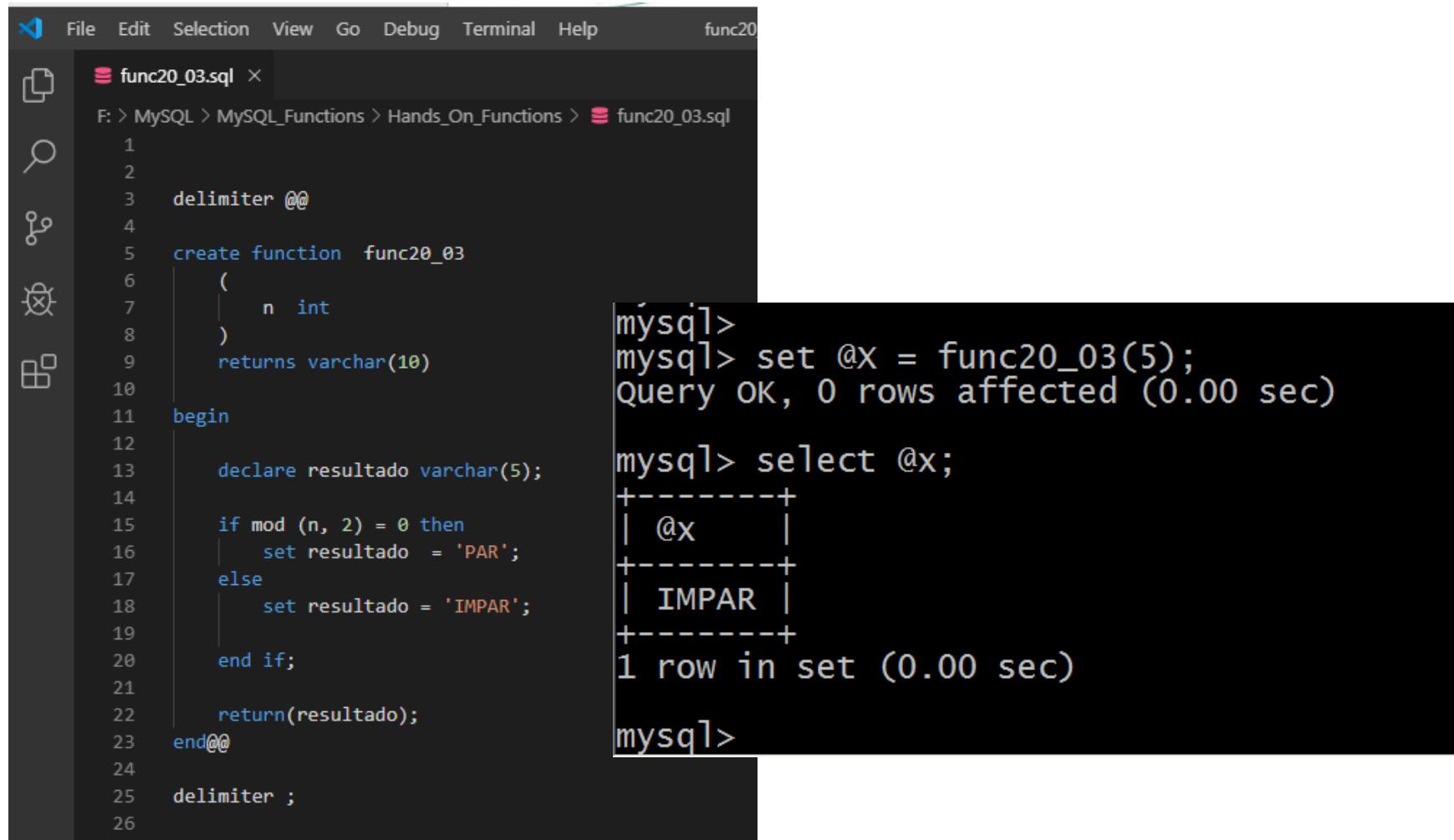
Uma **Function** também pode utilizar o conceito de bloco de comandos.

```
DELIMITER $$  
CREATE FUNCTION fn_DobraSomaTiposBloco(varValorA INT,  
                                         varValorB DECIMAL(6,2))  
RETURNS DECIMAL(6,2)  
BEGIN  
    DECLARE varSomador DECIMAL(6,2) DEFAULT 0;  
  
    SET varSomador = varValorA + varValorB;  
    SET varSomador = varSomador * 2;  
  
    RETURN varSomador;  
  
END$$  
DELIMITER ;
```

```
SELECT fn_DobraSomaTiposBloco(2, 3.54);
```

Chamada de Functions

Uma **function** pode ser chamada especificando-se seu nome e lista de parâmetros em uma expressão do tipo apropriado de dados que possa ser usado.



The screenshot shows the MySQL Workbench interface. On the left is a sidebar with icons for File, Edit, Selection, View, Go, Debug, Terminal, and Help. The main area has a title bar "func20" and a file list "func20_03.sql". The code editor contains the following SQL script:

```

1
2
3  delimiter @@
4
5  create function  func20_03
6  (
7      n  int
8  )
9  returns varchar(10)
10
11 begin
12
13     declare resultado varchar(5);
14
15     if mod (n, 2) = 0 then
16         set resultado = 'PAR';
17     else
18         set resultado = 'IMPAR';
19
20     end if;
21
22     return(resultado);
23 end@@
24
25 delimiter ;

```

To the right, the terminal window shows the execution of the function:

```

mysql>
mysql> set @x = func20_03(5);
Query OK, 0 rows affected (0.00 sec)

mysql> select @x;
+-----+
| @x   |
+-----+
| IMPAR |
+-----+
1 row in set (0.00 sec)

mysql>

```

User Defined Function

Exercício 1

Desenvolva uma function chamada **fn_CalcularReajuste**, esta função deve receber como parâmetro um valor que represente o código de um funcionário e outro parâmetro que representa o **percentual de reajuste** a ser atribuído a este funcionário.

Atenção: Esta função deve retornar o salário atual do funcionário acrescido do reajuste.

Após concluir o desenvolvimento desta function, teste seu funcionamento em uma instrução UPDATE.

User Defined Function

Exercício 1 *** RESOLUÇÃO ***

```
DELIMITER $$  
CREATE FUNCTION fn_CalcularReajuste(varIdFunc INT,  
                                     varPorcent DECIMAL(5,2))  
RETURNS DECIMAL(8,2)  
NOT DETERMINISTIC  
BEGIN  
    DECLARE varSalario DECIMAL(8,2);  
  
    SELECT f.salario INTO varSalario  
    FROM funcionario f  
    WHERE f.idfuncionario = varIdFunc;  
  
    RETURN (varSalario + ((varSalario * varPorcent) / 100));  
  
END$$  
DELIMITER ;
```

User Defined Function

Exercício 1 *** RESOLUÇÃO ***

```
SELECT f.idFuncionario, f.nome, f.salario
FROM funcionario f
WHERE f.idfuncionario = 7;
```



	idFuncionario	nome	salario
▶	7	Julio Monteiro	1875.00

```
UPDATE funcionario
SET salario = fn_CalcularReajuste(idfuncionario, 10)
WHERE idfuncionario = 7;
```

```
SELECT f.idFuncionario, f.nome, f.salario
FROM funcionario f
WHERE f.idfuncionario = 7;
```



	idFuncionario	nome	salario
▶	7	Julio Monteiro	2062.50

User Defined Function

Exercício 2

- a) Desenvolva uma function chamada **fn_MediaSalarialDepart**, esta função deve receber como parâmetro um valor que represente o código de um departamento.

Internamente a função deve calcular a média salarial do departamento e retornar seu valor.

Atenção: O retorno de sua função deve ser do tipo decimal;

User Defined Function

Exercício 2

- b) Após concluir e testar a função **fn_MediaSalarialDepart**, você deve desenvolver uma Stored Procedure chamada **sp_SalariosSuperioresMedia**.

Esta procedure não deve utilizar nenhum parâmetro de entrada ou saída, a única tarefa desta procedure é exibir uma listagem dos funcionários que recebem um salário maior do que a média do salário de seu departamento.

Essa listagem deve conter:

- ✓ Código do departamento;
- ✓ Nome do departamento;
- ✓ Código do funcionário;
- ✓ Nome do funcionário;
- ✓ Salário do funcionário;
- ✓ Quanto a mais o funcionário recebe comparado com a média de seu departamento.

User Defined Function

Exercício 2 – a) *** RESOLUÇÃO ***

```
SELECT f.idDepartamento, d.nome, AVG(f.salario)
FROM funcionario f
INNER JOIN departamento d USING(idDepartamento)
GROUP BY f.idDepartamento
```

	idDepartamento	nome	AVG(f.salario)
▶	1	Suporte Técnico	2780.000000
	2	Infraestrutura	4330.000000
	3	Segurança da Informação	3475.100000
	4	Desenvolvimento	4532.675000

```
SELECT AVG(f.salario)
FROM funcionario f
WHERE f.idDepartamento = 3;
```



	AVG(f.salario)
▶	3475.100000

User Defined Function

Exercício 2 – a) *** RESOLUÇÃO ***

```
DELIMITER $$  
CREATE FUNCTION fn_MediaSalarialDepart(varIdDepart INT)  
RETURNS DECIMAL(8,2)  
NOT DETERMINISTIC  
BEGIN  
    ....  
    RETURN ????;  
END$$  
DELIMITER ;
```



```
SELECT AVG(f.salario)  
FROM funcionario f  
WHERE f.idDepartamento = 3;
```



AVG(f.salario)
3475.100000

User Defined Function

Exercício 2 – a) *** RESOLUÇÃO ***

```
DELIMITER $$  
CREATE FUNCTION fn_MediaSalarialDepart(varIdDepart INT)  
RETURNS DECIMAL(8,2)  
NOT DETERMINISTIC  
BEGIN  
  
    SELECT AVG(f.salario)  
    FROM funcionario f  
    WHERE f.idDepartamento = 3;  
  
    RETURN ???;  
END$$  
DELIMITER ;
```



User Defined Function

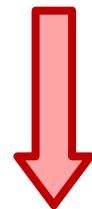
Exercício 2 – a) *** RESOLUÇÃO ***

```
DELIMITER $$  
CREATE FUNCTION fn_MediaSalarialDepart(varIdDepart INT)  
RETURNS DECIMAL(8,2)  
  
BEGIN  
  
    DECLARE varMediaSalario DECIMAL(8,2) DEFAULT 0;  
  
    SELECT AVG(f.salario) INTO varMediaSalario  
    FROM funcionario f  
    WHERE f.idDepartamento = varIdDepart;  
  
    RETURN varMediaSalario;  
END$$  
DELIMITER ;
```

User Defined Function

Exercício 2 – a) *** RESOLUÇÃO ***

```
SELECT fn_MediaSalarialDepart(3);
```



	fn_MediaSalarialDepart3(3)
▶	3475.10

User Defined Function

Exercício 2 – b) *** RESOLUÇÃO ***

```
DELIMITER $$  
CREATE PROCEDURE sp_SalariosSuperioresMedia()  
BEGIN  
    SELECT f.iddepartamento, d.nome,  
           f.idFuncionario, f.nome,  
           f.salario,  
           (f.salario - fn_MediaSalarialDepart(f.idDepartamento)) DifSalarial  
    FROM Funcionario f  
    INNER JOIN departamento d USING(idDepartamento)  
    WHERE f.salario >= fn_MediaSalarialDepart(f.idDepartamento)  
    ORDER BY f.iddepartamento;  
END$$  
DELIMITER ;
```

```
CALL sp_SalariosSuperioresMedia();
```

User Defined Function

Exercício 3

Desenvolva uma function chamada `fn_CalculoInss`, esta função deve receber como parâmetro um valor que represente o código de um funcionário;

Esta função deve retornar o valor a ser recolhido ao INSS com base no salário do funcionário;

Atenção: O cálculo do recolhimento do INSS deve respeitar os índices estabelecidos conforme a seguinte tabela:

SALÁRIO DE CONTRIBUIÇÃO (R\$)	ALÍQUOTA PARA FINS DE RECOLHIMENTO AO INSS
até 1.659,38	8%
de 1.659,39 até 2.765,66	9%
de 2.765,67 até 5.531,31	11%

User Defined Function

Exercício 3 *** RESOLUÇÃO ***

```
DELIMITER $$  
CREATE FUNCTION fn_CalcInss (varIdFunc INT) RETURNS DECIMAL(8,2)  
BEGIN  
  
    DECLARE varAliquota DECIMAL(5,2);  
    DECLARE varSalarioBase DECIMAL(8,2) DEFAULT 0;  
  
    SELECT f.salario INTO varSalarioBase  
    FROM funcionario f WHERE f.idFuncionario = varIdFunc;  
  
    IF varSalarioBase > 5531.31 THEN  
        SET varSalarioBase = 5531.31;  
    END if;  
  
    CASE  
        WHEN (varSalarioBase <= 1659.38) THEN SET varAliquota = 0.08;  
        WHEN (varSalarioBase <= 2765.66) THEN SET varAliquota = 0.09;  
        WHEN (varSalarioBase <= 5531.31) THEN SET varAliquota = 0.11;  
    END CASE;  
  
    RETURN varSalarioBase * varAliquota;  
  
END$$  
DELIMITER ;
```