

# LAB 04

Ícaro Lima Kuchanovicz

Caroline Assis

Ary Farah

Adriano Vale

## INSERT: DEFAULT e NULL

A)

```
CREATE DATABASE Lab_04;

USE Lab_04;

CREATE TABLE Tab_Depto(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  Nome VARCHAR(60) NOT NULL DEFAULT ('Vendas'),
  Localizacao VARCHAR(60) DEFAULT ('Bloco A'),
  Sala CHAR(3) NOT NULL,
  Fone VARCHAR(20)
);
```

B)

```
INSERT Tab_Depto (Sala, Fone) VALUES ('80', '(41)3021-4040');
INSERT Tab_Depto (Sala) VALUES ('100');
INSERT Tab_Depto (Localizacao, Sala) VALUES (NULL, '200');
INSERT Tab_Depto (Sala) VALUES (NULL);
INSERT Tab_Depto (Localizacao, Sala) VALUES (NULL, '300');
SELECT * FROM Tab_Depto;
```

Neste exercício, a criação da Tab\_Depto define seus campos com configuração para NULL, NOT NULL e DEFAULT

### Pergunta 1

Em b.1), como é feito o tratamento de campos omitidos?

- O campo *ID* foi preenchido pelo `auto_increment`
- O campo *Nome* foi preenchido pelo `default ('Vendas')`
- O campo *Localização* foi preenchido pelo `default ('Bloco A')`

### Pergunta 2

Em b.2), como é feito o tratamento de campos omitidos?

- O campo *ID* foi preenchido pelo `auto_increment`
- O campo *Nome* foi preenchido pelo `default ('Vendas')`
- O campo *Localização* foi preenchido pelo `default ('Bloco A')`
- O campo *Fone* foi preenchido com `null`

### Pergunta 3

Em b.3), como é feito o tratamento do **NULL**?

- Caso deixasse o campo *Localização* omitido, ele seria preenchido com o `default ('Bloco A')`, porém como eu quero que fique com **NULL**, preciso informar que é **NULL**

### Pergunta 4

Em b.4), o que acontece neste **INSERT** do **NULL**?

- O código tenta colocar NULL em todos os campos, porém acontece um erro, porque o campo *Sala* tem uma constraint **NOT NULL** SEM um default declarado, ou seja, ela PRECISA que um valor seja atribuído a ela

## Pergunta 5

Em b.5), o que acontece neste **INSERT** do **NULL**?

- Igual ao b.3), caso deixasse o campo *Localização* omitido, ele seria preenchido com o `default ('Bloco A')`, porém como eu quero que fique com **NULL**, preciso informar que é **NULL**

1. Em b.5), como ficou povoada a **Tab\_Depto**?

	ID	Nome	Localizacao	Sala	Fone
▶	1	Vendas	Bloco A	80	(41)3021-4040
	2	Vendas	Bloco A	100	NULL
	3	Vendas	NULL	200	NULL
	4	Vendas	NULL	300	NULL
*	NULL	NULL	NULL	NULL	NULL

# AÇÕES PARA MANTER A IR

## CASCADE

(A)

```
CREATE TABLE Editora(
ID_edit INT AUTO_INCREMENT PRIMARY KEY, -- Tabela PAI
Nome_Edit VARCHAR(60) NOT NULL,
Cidade VARCHAR(60) NOT NULL,
Estado CHAR(2) NOT NULL,
Pais VARCHAR(50) NOT NULL
```

```
);
```

```
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Editora AAA', 'São Paulo', 'SP', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Editora Sul', 'Porto Alegre', 'RS', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('LTC', 'São Paulo', 'SP', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('CENGAGE', 'Rio de Janeiro', 'RJ', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Três Estrelas', 'Alagoas', 'CE', 'Brasil');
```

## (B)

```
CREATE TABLE Autor(
ID_Autor INT AUTO_INCREMENT PRIMARY KEY,
Nome_Autor VARCHAR(60) NOT NULL,
Dt_Nasc DATE NOT NULL,
fk_ID_Edit INT NULL
);
```

```
ALTER TABLE Autor ADD CONSTRAINT FK_Autor_Editora FOREIGN KEY(fk_ID_edit)
REFERENCES Editora (ID_edit)
ON UPDATE CASCADE
ON DELETE CASCADE ;
ALTER TABLE Autor AUTO_INCREMENT = 100;
```

```
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('José', '1956-09-08', 1);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Maria', '1975-04-18', 2);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Antônia', '1954-12-10', 3);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Armínio', '1976-07-28', 5);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Luiza', '1945-11-09', 5);
```

## EXECUTE OS CÓDIGOS

### (C)

```
SELECT * FROM Editora;
SELECT * FROM Autor;
```

### (D)

```
SELECT * FROM Editora;
SELECT * FROM Autor;
```

```
UPDATE Editora
  SET ID_edit = 50
  WHERE ID_edit = 5;

SELECT * FROM Editora;
SELECT * FROM Autor;
```

```
DELETE FROM Editora
  WHERE ID_edit = 1;

SELECT * FROM Editora
SELECT * FROM Autor
```

## Pergunta 1

Em c), qual foi o resultado obtido nas tabelas Editora e Autor, após o UPDATE executado? Por que isso ocorreu?

- As **duas** tabelas foram atualizadas na coluna ID\_edit mesmo com o comando sendo referenciado apenas a tabela *Editora* por causa do **ON UPDATE CASCADE**, que faz com que qualquer mudança na tabela seja “distribuído” para as outras tabelas que tem uma relação com ela

## Pergunta 2

Em d), qual foi o resultado obtido nas tabelas Editora e Autor, após o DELETE executado? Por que isso ocorreu?

- As **duas** tabelas foram atualizadas excluindo a linha que tinha o ID\_edit = 1.
- Apesar do comando sendo referenciado apenas a tabela *Editora*, por causa do **ON UPDATE CASCADE**, a tabela filho (que tem uma chave estrangeira sendo referenciada a tabela pai) também recebeu essa mudança

## RESTRICT

**(E)**

```
UPDATE Editora SET ID_edit = 5 WHERE ID_edit = 50;

DROP TABLE Autor;

CREATE TABLE Autor(
ID_Autor INT AUTO_INCREMENT PRIMARY KEY, -- Tabela FILHO
Nome_Autor VARCHAR(60) NOT NULL,
Dt_Nasc DATE NOT NULL,
fk_ID_Edit INT NULL
);

ALTER TABLE Autor ADD CONSTRAINT FK_Autor_Editora FOREIGN KEY(fk_ID_edit) REFERENCES Editora (ID_edit) ON UPDATE RESTRICT ON DELETE RESTRICT ;

ALTER TABLE Autor AUTO_INCREMENT = 100;

INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('José', '1956-09-08', 1);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Maria', '1975-04-18', 2);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Antônia', '1954-12-10', 3);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Arminio', '1976-07-28', 5);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Luiza', '1945-11-09', 5);
```

## EXECUTE OS CÓDIGOS

**(F)**

```
SELECT * FROM Editora;
SELECT * FROM Autor;

UPDATE Editora
SET ID_edit = 50
WHERE ID_edit = 5;

SELECT * FROM Editora;
SELECT * FROM Autor;
```

**(G)**

```
SELECT * FROM Editora;
SELECT * FROM Autor;

DELETE FROM Editora
WHERE ID_edit = 1;

SELECT * FROM Editora;
SELECT * FROM Autor;
```

## Pergunta 1

Em f), qual foi o resultado obtido nas tabelas Editora e Autor, após o UPDATE executado? Por que isso ocorreu?

- Deu um erro
- O erro aconteceu porque na tabela foi adicionado o `ON UPDATE RESTRICT`. Esse comando bloqueia a modificação quando você tenta alterar alguma coluna que está relacionada com outra tabela

## Pergunta 2

Em g), qual foi o resultado obtido nas tabelas Editora e Autor, após o DELETE executado? Por que isso ocorreu?

- Aconteceu o mesmo erro do código (f)
- O erro aconteceu porque na tabela foi adicionado o `ON DELETE RESTRICT`. Esse comando bloqueia a modificação quando você tenta deletar alguma coluna que está relacionada com outra tabela

## SET NULL

### (H)

```
UPDATE Editora
SET ID_edit = 5
WHERE ID_edit = 50;
DROP TABLE Autor;

CREATE TABLE Autor(
ID_Autor INT AUTO_INCREMENT PRIMARY KEY,
Nome_Autor VARCHAR(60) NOT NULL,
Dt_Nasc DATE NOT NULL,
fk_ID_Edit INT NULL
);
```

```

ALTER TABLE Autor ADD CONSTRAINT FK_Autor_Editora FOREIGN KEY(fk_ID_edit)
REFERENCES Editora (ID_edit)
ON UPDATE SET NULL
ON DELETE SET NULL;

ALTER TABLE Autor AUTO_INCREMENT = 100;

INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('José', '1956-09-08', 1);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Maria', '1975-04-18', 2);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Antônia', '1954-12-10', 3);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Arminio', '1976-07-28', 5);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Luiza', '1945-11-09', 5);

```

## EXECUTE OS CÓDIGOS

(I)

```

SELECT * FROM Editora;
SELECT * FROM Autor;

UPDATE Editora
  SET ID_edit = 50
  WHERE ID_edit = 5;

SELECT * FROM Editora;
SELECT * FROM Autor;

```

(J)

```

SELECT * FROM Editora;
SELECT * FROM Autor;

DELETE FROM Editora
WHERE ID_edit = 1;

SELECT * FROM Editora;
SELECT * FROM Autor;

```

### Pergunta 1

Em i), qual foi o resultado obtido nas tabelas Editora e Autor, após o UPDATE executado? Por que isso ocorreu?

- O comando foi executado, porém como tem o **ON UPDATE SET NULL**, no momento em que o comando **UPDATE** foi executado, na tabela *Editora* (pai) as colunas foram atualizadas, porém na tabela *Autor* (filho), aquelas linhas que estavam relacionadas a tabela pai foram automaticamente preenchidas com **NULL**



## Pergunta 2

Em j), qual foi o resultado obtido nas tabelas Editora e Autor, após o DELETE executado? Por que isso ocorreu?

- O comando foi executado, porém como tem o `ON DELETE SET NULL`, no momento em que o comando `DELETE` foi executado, na tabela *Editora* (pai) as colunas foram atualizadas, porém na tabela *Autor* (filho), aquelas linhas que estavam relacionadas a tabela pai foram automaticamente preenchidas com **NULL**

## INSERT em VIEWS

(a)

```
CREATE TABLE Tab_Um (  
  ID_um INT PRIMARY KEY NOT NULL,  
  col_1 CHAR(3) NOT NULL  
);  
  
CREATE TABLE Tab_Dois (  
  fk_ID_um INT PRIMARY KEY NOT NULL,  
  col_2 CHAR(3) NOT NULL,  
  FOREIGN KEY (FK_ID_um)  
  REFERENCES Tab_Um (ID_um)  
);  
  
CREATE VIEW JuntaUmDois AS (  
  SELECT ID_um, col_1, fk_ID_um, col_2  
  FROM Tab_Um JOIN Tab_Dois  
  ON (Tab_Um.ID_um = Tab_Dois.fk_ID_um)  
);
```

(b)

```
-- 1º. INSERT  
INSERT Tab_Um (ID_um, col_1) VALUES (5, 'AAA');  
SELECT * FROM JuntaUmDois;  
SELECT * FROM Tab_Um;
```

```

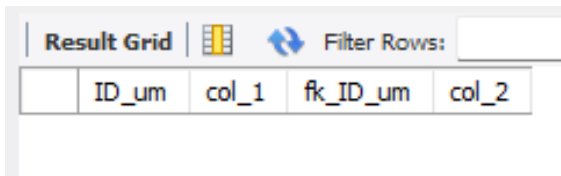
SELECT * FROM Tab_Dois;

-- 2º. INSERT
INSERT Tab_Dois(fk_ID_um, col_2) VALUES (5, 'XXX');
SELECT * FROM JuntaUmDois;
SELECT * FROM Tab_Um;
SELECT * FROM Tab_Dois;

```

## Pergunta 1

Em b), após o 1º. INSERT, o que foi exibido na VIEW? Por que esse resultado foi apresentado?

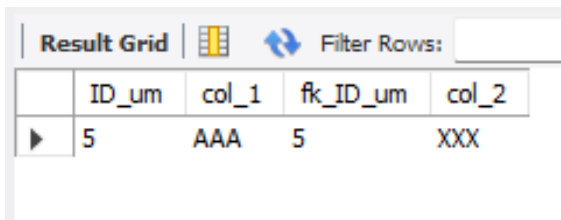


	ID_um	col_1	fk_ID_um	col_2
--	-------	-------	----------	-------

- O resultado está vazio porque a view mostra apenas os resultados correspondentes entre as tabelas (que nesse caso não tem)

## Pergunta 2

Em b), após o 2º. INSERT, o que foi exibido na VIEW? Por que esse resultado foi apresentado?



	ID_um	col_1	fk_ID_um	col_2
▶	5	AAA	5	XXX

- O resultado foi onde os dados são correspondentes nas duas tabelas, nesse caso onde o ID é igual a 5

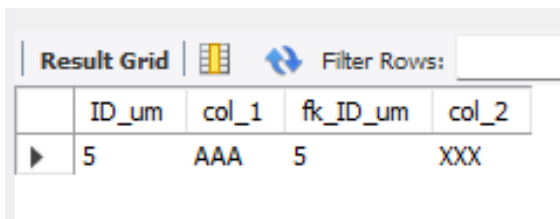
(C)

```
-- 1º. INSERT
INSERT JuntaUmDois (ID_Um, col_1)
VALUES (10, 'BBB');
SELECT * FROM JuntaUmDois;
SELECT * FROM Tab_Um;
SELECT * FROM Tab_Dois;

-- 2º. INSERT
INSERT JuntaUmDois(fk_ID_um, col_2)
VALUES (20, 'YYY');
SELECT * FROM JuntaUmDois;
SELECT * FROM Tab_Um;
SELECT * FROM Tab_Dois;
```

## Pergunta 1

Em c), após o 1º. INSERT, o que foi exibido na VIEW? Por que esse resultado foi apresentado?



The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. The grid contains one row of data with the following values:

	ID_um	col_1	fk_ID_um	col_2
▶	5	AAA	5	XXX

- O resultado continua o mesmo porque foram adicionados valores nas colunas *ID\_Um* e *col\_1* e a view mostra uma relação entre as duas tabelas, ou seja, todas as colunas dessa view precisam estar preenchidas

## Pergunta 2

Em c), o 2º. INSERT na VIEW funcionou na Tab\_Dois? Por que?

- Não funcionou
- Não tem como adicionar um valor numa coluna que é chave estrangeira de outra tabela

### Pergunta 3

Em c), como devemos alterar o 2º. INSERT, para que ele funcione na Tab\_Dois e também seja exibido na VIEW?

- `INSERT JuntaUmDois(fk_ID_um, col_2) VALUES (10, 'YYY');`
- Troquei o valor do `fk_ID_um` para um valor que já existia na tabela pai (10), assim as duas chaves estão se relacionando e aparecem no resultado da view

### Pergunta 4

Após corrigir o 2º. INSERT, mostre como ficam preenchidas as Tab\_Um, Tab\_Dois e VIEW JuntaUmDois.

- Tab\_Um
- Tab\_Dois
- JuntaUmDois

Result Grid		
ID_um	col_1	
5	AAA	
10	BBB	
NULL	NULL	

Result Grid		
fk_ID_um	col_2	
5	XXX	
10	YYY	
NULL	NULL	

Result Grid				
ID_um	col_1	fk_ID_um	col_2	
5	AAA	5	XXX	
10	BBB	10	YYY	

## VARIÁVEIS

(a)

```

USE LAB_04;
CREATE TABLE Empresa (
  ID INT PRIMARY KEY AUTO_INCREMENT,
  Nome VARCHAR(20),
  Atuacao VARCHAR(50),
  Cidade VARCHAR(20),
  Estado VARCHAR(2)
);

INSERT Empresa (Nome, Atuacao, Cidade, Estado) VALUES
('ACME Corp.', 'Cartoons', 'São Paulo', 'SP'),
('Estrela Ltda.', 'Transporte passageiros', 'Campinas', 'SP'),
('Aurora', 'Panificadora', 'Belo Horizonte', 'MG'),
('Azul', 'Aviação', 'São Paulo', 'SP'),
('Leão Ltda.', 'Bebidas', 'Curitiba', 'PR'),
('Petit S.A.', 'Queijos e frios', 'Uberlândia', 'MG'),
('Barreados Corp.', 'Alimentos congelados', 'Morretes', 'PR');

SELECT * FROM Empresa;

```

**(b)**

```

CREATE TABLE Estoque (
  ID INT PRIMARY KEY AUTO_INCREMENT,
  Nome VARCHAR(20),
  Qtde INT DEFAULT 10,
  ValUnit DECIMAL(10,2)
);

INSERT Estoque (Nome, Qtde, ValUnit)
VALUES
('caderno', 200, 15.00),
('borracha', 50, 6.50),
('caneta', 300, 5.50),
('régua 30cm', 80, 10.00),
('lápiz', 500, 4.00),
('bloco A4', 35, 18.45);

SELECT * FROM Estoque;

```

**(c)**

```
SET @nome_produto = 'none';
SET @total_produtos = -1;

SELECT nome INTO @nome_produto
FROM Estoque
WHERE ID = 3;

SELECT COUNT(*) INTO @total_produtos
FROM Estoque;

SELECT @nome_produto AS 'Produto com ID = 3';
SELECT @total_produtos AS 'Total de Produtos Cadastrados';
```

## Pergunta 1

Nas linhas 1. e 2., o que acontece se não inicializarmos as variáveis?

- Dará erro nas próximas linhas onde essas variáveis serão usadas

## Pergunta 2

É preciso executar de uma única vez todos os comandos do script apresentado, para exibir o conteúdo final das variáveis de sessão do exemplo? Por que?

- Não
- As variáveis ficam “salvas” durante a execução do SQL Server. Então contanto que os comandos sejam executados na ordem certa, pode executar um de cada vez

(d)

```
DELIMITER $$
CREATE PROCEDURE proc_demo1()
BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE output VARCHAR(100) DEFAULT 'Saída = ';
    WHILE i < 10 DO
        SET output = CONCAT(output, i , ' , ');
        SET i = i + 1;
    END WHILE;
    SELECT output;

END $$
DELIMITER ;

CALL proc_demo1();
```

## Pergunta 1

Em que linhas do código estão os delimitadores do bloco de comandos da procedure?

- [Linha 4 → Começa](#)
- [Linha 12 → Acaba](#)

## Pergunta 2

O que é feito nas linhas 5. e 6.?

- [Declara e inicializa as variáveis locais](#)

## Pergunta 3

Quais os delimitadores do laço de repetição WHILE?

- [Linha 7 \( WHILE \)](#)
- [Linha 10 \( END WHILE \)](#)

## Pergunta 4

O que é feito nas linhas 8., .9 e .15?

- Linha 8
  - Atualiza a variável *output* com uma concatenação
  - Pega o valor atual da variável *output*, acrescenta a variável *i* com o separador vírgula
- Linha 9
  - Pega a variável *i* e acrescenta + 1
- Linha 10
  - É como se chamasse a “função `proc_demo1()`”

## IF / CASE

(a)

```
SELECT IF (WEEKDAY(NOW()) IN (5, 6), 'É FIM de semana', 'É DIA de semana') AS 'DIA DE HOJE';
```

```
SELECT IF (WEEKDAY('2023-09-24') IN (5, 6), 'É FIM de semana', 'É DIA de semana') AS '24/09/2023';
```

(b)

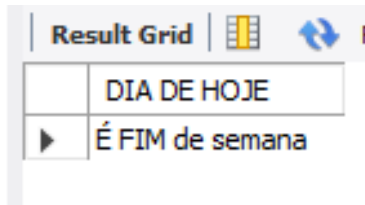
```
SELECT ID AS 'Código', Nome, Qtde AS 'Quantidade',  
IF (Qtde < 100, 'Baixo (menor que 100)', 'Em boa quantidade') AS 'Nível Estoque'  
FROM Estoque;
```

## Pergunta 1



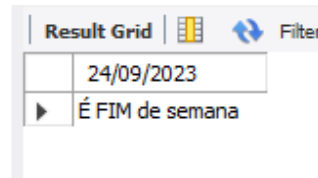
Em a) qual a diferença entre os comandos SELECT? Qual o resultado?

### 1º SELECT



	DIA DE HOJE
▶	É FIM de semana

### 2º SELECT

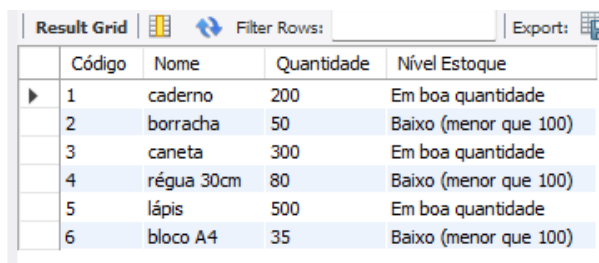


	24/09/2023
▶	É FIM de semana

- 1º SELECT retorna se é dia de semana ou final de semana com base no dia atual. Já o 2º retorna isso com base numa data fixa

## Pergunta 2

Em b) apresente e explique o que aparece na coluna 'Nível Estoque' do SELECT?



	Código	Nome	Quantidade	Nível Estoque
▶	1	caderno	200	Em boa quantidade
	2	borracha	50	Baixo (menor que 100)
	3	caneta	300	Em boa quantidade
	4	régua 30cm	80	Baixo (menor que 100)
	5	lápiz	500	Em boa quantidade
	6	bloco A4	35	Baixo (menor que 100)

- Dentro do SELECT tem um IF que confere o valor da coluna *Quantidade*, caso seja maior do que 100 recebe o valor 'Em boa quantidade' e caso seja menor que 100 recebe 'Baixo (menor que 100)'

(c)

```
INSERT INTO Estoque (Nome, ValUnit) VALUES ('cola bastão', 15.00);

INSERT INTO Estoque (Nome, Qtde, ValUnit) VALUES ('tesoura', NULL, 15.00);

SELECT ID AS 'Código', Nome, Qtde AS 'Quantidade',
CASE
  WHEN Qtde < 100 THEN 'BAIXO'
  WHEN Qtde BETWEEN 100 AND 300 THEN 'OK'
  WHEN Qtde > 300 THEN 'ALTO'
  ELSE 'DESCONHECIDO'
END AS 'Nível Estoque'
FROM Estoque
ORDER BY Nome;
```

## Pergunta 1

No 1º. INSERT, não foi especificado um valor de Qtde. Qual o Nível de Estoque apresentado no SELECT? Por que esse valor foi exibido?

- Foi apresentado o valor '10'
- A coluna quantidade tem o default com o valor '10'. E como não foi atribuído nenhum valor na coluna quantidade para o 1º insert, o valor atribuído foi o default

## Pergunta 2

No 2º. INSERT, foi especificado que de Qtde = NULL. Qual o Nível de Estoque apresentado no SELECT? Por que esse valor foi exibido?

- O nível de estoque apresentado foi “**null**” porque no insert o valor **null** foi atribuído na coluna *quantidade*

### Pergunta 3

Quando a opção ELSE do CASE é executada?

- Quando o valor da coluna quantidade não for nem menor nem igual nem maior que cem
- No caso do último `insert`, o ELSE foi executado porque na coluna *quantidade* estava o valor **NULL**

### Pergunta 4

Apresente o resultado do SELECT.

	Código	Nome	Quantidade	Nível Estoque
▶	6	bloco A4	35	BAIXO
	2	borracha	50	BAIXO
	1	caderno	200	OK
	3	caneta	300	OK
	7	cola bastão	10	BAIXO
	5	lápiz	500	ALTO
	4	régua 30cm	80	BAIXO
	8	tesoura	NULL	DESCONHECIDO

## COMPARAÇÃO

(a)

```
SELECT nome, Atuacao, Cidade, Estado FROM Empresa
WHERE
((Cidade <> 'São Paulo' AND Estado <> 'SP')
OR
(Cidade <> 'Morretes' AND Estado <> 'P
```

(b)

```
SELECT nome, Atuacao, Cidade, Estado FROM Empresa
WHERE NOT (
((Cidade = 'São Paulo' AND Estado = 'SP')
OR
```

```
R')  
);
```

```
(Cidade = 'Morretes' AND Estado = 'PR'))  
);
```

Encontrar as empresas que não estão nem em São Paulo, SP nem em Morretes, PR.

## Pergunta 1

Apresente o resultado de cada SELECT.

### 1° SELECT

nome	Atuacao	Cidade	Estado
ACME Corp.	Cartoons	São Paulo	SP
Estrela Ltda.	Transporte passageiros	Campinas	SP
Aurora	Panificadora	Belo Horizonte	MG
Azul	Aviação	São Paulo	SP
Leão Ltda.	Bebidas	Curitiba	PR
Petit S.A.	Queijos e frios	Uberlândia	MG
Barreados Corp.	Alimentos congelados	Morretes	PR

### 2° SELECT

nome	Atuacao	Cidade	Estado
Estrela Ltda.	Transporte passageiros	Campinas	SP
Aurora	Panificadora	Belo Horizonte	MG
Leão Ltda.	Bebidas	Curitiba	PR
Petit S.A.	Queijos e frios	Uberlândia	MG

## Pergunta 2

Qual consulta de SELECT satisfaz o enunciado? Por que?

- O 2° select
- Ele conseguiu trazer como resultado as empresas que não são nem de São Paulo, nem de Morretes

# FUNÇÕES MATEMÁTICAS

(a)

```
SET @angle = PI()/4; -- 45° em rad
SELECT CONCAT( 'O SENO do ângulo: ' ,
  CONVERT(ROUND(@angle,3), CHAR) ,
  ' rad = ' ,
  CONVERT(ROUND(SIN(@angle),3), CHAR)) AS 'SENO 45° (ou PI/4 rad)';
```

## Pergunta 1

Qual é a variável utilizada no exemplo e como ela foi definida?

- Foi utilizada a variável de sessão **@angle** iniciada com o valor de  $\text{PI} / 4$

## Pergunta 2

O que faz a função `PI()`?

- Retorna o valor de  $\text{PI}$

## Pergunta 3

O que faz a função `CONCAT()`?

- Concatena (junta) várias strings em uma única string

## Pergunta 4

O que faz a função `CONVERT()`?

- Converte o tipo, nesse caso transformando um número em uma string

## Pergunta 5

Apresente o resultado do comando `SELECT` do exemplo

Result Grid		Filter Rows:
	SENO 45° (ou PI/4 rad)	
	O SENO do ângulo: 0.785 rad = 0.707	

(b)

```
DROP PROCEDURE IF EXISTS proc_demo2;

DELIMITER \\\
CREATE PROCEDURE proc_demo2(IN angle FLOAT, OUT output VARCHAR (100))
BEGIN
    SET output = '';
    SET output = CONCAT (output,
        ' [ ANGULO_GRAUS = ', CONVERT(ROUND(@angle * 180 / PI (), 3), CHAR), ']',
        ' [ ANGULO_RAD = ', CONVERT(ROUND(@angle, 3 ), CHAR), ']',
        ' [ SENO = ', CONVERT(ROUND(SIN(@angle),3 ), CHAR), ']',
        ' [ COSSENO = ', CONVERT(ROUND(COS(@angle),3 ), CHAR), ']',
        ' [ TANGENTE = ', CONVERT(ROUND(TAN(@angle),3 ), CHAR), ']');
END \\\
DELIMITER ;
```

(c)

```
SET @angle = PI()/3;
SET @resp = '';
CALL proc_demo2(@angle, @resp);
SELECT @resp AS 'RESPOSTA';

SET @angle = PI()/4;
SET @resp = '';
CALL proc_demo2(@angle, @resp);
SELECT @resp AS 'RESPOSTA';

SET @angle = PI()/6;
SET @resp = '';
CALL proc_demo2(@angle, @resp);
SELECT @resp AS 'RESPOSTA';
```

## Pergunta 1

Em b) Para que servem os comandos das linhas 3. e 14.?

- Linha 3
  - “Troca” o delimitador de código: a função do `;` passa a ser agora do `//`
- Linha 14
  - Volta com o delimitador de código pro padrão (`;`)

## Pergunta 2

Quais são e como são definidos os parâmetros da procedure `proc_demo2()`?

- Os parâmetros são `IN angle FLOAT` e `OUT output VARCHAR (100)`
- O parâmetro `IN angle FLOAT` recebe um valor ai iniciar a procedure. O valor dessa variável vai ser usado para realizar os calculos no meio da procedure
- O parâmetro `OUT output VARCHAR (100)` é onde será armazenado o resultado dessa procedure

## Pergunta 3

O que está sendo feito na atribuição que inicia na linha 7.?

- Atribuindo o resultado da conta á variável `output`

## Pergunta 4

Em c) apresente e explique o resultado o de cada um dos 3 conjuntos de comandos.

- **CONJUNTO 1**

- Calculou os resultados com base no ângulo dado:  $\pi/3$

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	RESPOSTA			
▶	[ ANGULO_GRAUS = 60 ] [ ANGULO_RAD = 1.047 ] [ SENO = 0.866 ] [ COSSENO = 0.5 ] [ TANGENTE = 1.732 ]			

- **CONJUNTO 2**

- Calculou os resultados com base no ângulo dado:  $\pi/4$

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	RESPOSTA			
▶	[ ANGULO_GRAUS = 45 ] [ ANGULO_RAD = 0.785 ] [ SENO = 0.707 ] [ COSSENO = 0.707 ] [ TANGENTE = 1 ]			

- **CONJUNTO 3**

- Calculou os resultados com base no ângulo dado:  $\pi/6$

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	RESPOSTA			
▶	[ ANGULO_GRAUS = 30 ] [ ANGULO_RAD = 0.524 ] [ SENO = 0.5 ] [ COSSENO = 0.866 ] [ TANGENTE = 0.577 ]			

## STORES PROCEDURES RECURSIVAS

(a)

```
DROP PROCEDURE IF EXISTS fatorial;  
  
DELIMITER //
```



```

CREATE PROCEDURE fatorial(IN param INT, OUT total INT)
BEGIN
    DECLARE param_menos INT DEFAULT NULL ;
    DECLARE tmp_total INT DEFAULT -1;
    SET @@max_sp_recursion_depth = 50;
    IF (param IS NULL) OR (param < 0) OR (param > 12)
        THEN SET total = -1;
    ELSEIF (param = 0) OR (param = 1)
        THEN SET total = 1;
    ELSE
        SET param_menos = param - 1;
        CALL fatorial(param_menos, tmp_total);
        IF (tmp_total = -1)
            THEN SET total = -1;
            ELSE SET total = tmp_total * param;
        END IF;
    END IF;
END //
DELIMITER ;

```

## Pergunta 1

O que é feito no comando da linha 8?

- Definindo como 50 a quantidade máxima que a procedure pode chamar ela própria

## Pergunta 2

Se o parâmetro de entrada param não tiver valor (= NULL) qual será o valor do parâmetro de saída total?

- -1

## Pergunta 3

Qual o valor máximo que podemos utilizar para calcular o fatorial, no exemplo passado

- 12

(b)

```
-- conjunto comandos 1:
SET @resp = -1;
CALL fatorial (0, @resp);
SELECT @resp;

-- conjunto comandos 2:
CALL fatorial (13, @resp);
SELECT @resp;

-- conjunto comandos 3:
CALL fatorial (4, @resp);
SELECT @resp;

-- conjunto comandos 4:
CALL fatorial (-4, @resp);
SELECT @resp;

-- conjunto comandos 5:
CALL fatorial (6, @resp);
SELECT @resp;
```

## Pergunta 1

Mostre o resultado da execução do conjunto de comandos, indicando qual o valor numérico que está sendo calculado o fatorial. Justifique, de acordo com a stored procedure, o resultado apresentado

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>• <b>CONJUNTO 1</b></li><li>• Valor = 0</li><li>• Dentro da procedure tem um IF pra caso o valor seja 0 retorne 1</li></ul> | <ul style="list-style-type: none"><li>• <b>CONJUNTO 2</b></li><li>• Valor = 13</li><li>• Como o valor era maior que 12, foi retornado -1</li></ul> |
|---|--|

	@resp
▶	1

	@resp
▶	-1

- **CONJUNTO 3**

- Valor = 4
- Calculou o fatorial de 4

	@resp
▶	24

- **CONJUNTO 4**

- Valor = -4
- Valor era negativo, retorna -1

	@resp
▶	-1

- **CONJUNTO 5**

- Valor = 6
- Calcula o fatorial de 6

	@resp
▶	720