# **Exercícios de Fixação**

# **(5) Funções do SQL Server**

# **(5.1) Casting and Conversion Functions:**

**1)** Assinale a resposta **CORRETA**:

(1) O CAST e CONVERT possuem funcionalidades semelhantes e a mesma sintaxe.

(2) Quando usamos o CAST e uma conversão falha não irá gerar um erro no código, mas se usar o TRY\_CAST vai gerar um erro sim.

(3) O resultado do comando abaixo é: ‘256’

DECLARE @VAR\_TESTE INT = 156

SELECT CAST(@VAR\_TESTE AS VARCHAR(10)) + '100'

(4) O resultado do comando abaixo é: 7.5

DECLARE @VAR\_TESTE INT = 10

SELECT CAST( ((@VAR\_TESTE \* 2 - 5) / 2 ) AS NUMERIC(9,2))

(5) O resultado do comando abaixo é:

SELECT ISNUMERIC(100.50) -- 1

SELECT ISNUMERIC('100.50') -- 0

a) F, F, F, V, F

b) F, F, F, F, F

c) F, F, V, V, V

d) V, F, F, F, F

e) V, F, F, V, V

**2)** O que podemos afirmar sobre o comando abaixo:

-- Definição da Tabela

CREATE TABLE Cliente (

Id\_Cliente INT IDENTITY(1,1) NOT NULL,

Nm\_Cliente VARCHAR(100) NOT NULL,

Dt\_Nascimento DATETIME NOT NULL,

PRIMARY KEY(Id\_Cliente)

)

-- Índice da Tabela

CREATE NONCLUSTERED INDEX SK01\_Dt\_Nascimento

ON Cliente(Dt\_Nascimento)

INCLUDE(Nm\_Cliente)

-- Fazendo um filtro na Tabela

SELECT \*

FROM Cliente

WHERE CAST(Dt\_Nascimento AS DATE) = '19890914'

a) O CAST não influencia na performance, pois não faz diferença se ele é utilizado no SELECT ou no WHERE.

b) O CAST não influencia na performance, pois a coluna “Dt\_Nascimento“ já possui um índice e a query irá utilizá-lo normalmente.

c) O CAST pode influenciar na performance, pois o índice que foi criado na coluna “Dt\_Nascimento” é do tipo nonclustered. Se ele fosse do tipo clustered não teria problema.

d) O CAST pode influenciar na performance, pois não é uma boa prática utilizar funções nas colunas da tabela no WHERE, visto que isso pode afetar o plano de execução da query.

e) Não sei Luiz, vou ver o Curso de Dicas de Performance do Fabiano Amorim pra aprender aqui.

# **(5.2) Date and Time Functions:**

**3)** Assinale a alternativa **CORRETA**:

(1) O comando abaixo irá acrescentar uma hora na data/hora atual.

SELECT GETDATE() + 1

(2) Os comandos abaixo retornam a hora e minuto da data/hora atual, respectivamente.

SELECT HOUR(GETDATE())

SELECT MINUTE(GETDATE())

(3) Com o comando DATEPART ( datepart , date) podemos retornar uma parte específica de uma data.

(4) Os comandos DATEADD e DATEDIFF possuem a mesma sintaxe, a diferença é que um soma e o outro faz a diferença entre as datas.

a) F, F, V, F

b) F, F, V, V

c) F, F, F, V

d) V, F, V, F

e) V, F, V, V

**4)** Assinale a alternativa **CORRETA**:

(1) O comando abaixo é inválido e irá gerar um erro, pois não é possível usar um valor negativo no DATEADD.

SELECT DATEADD(HOUR, -12, GETDATE())

(2) O comando abaixo é inválido e irá gerar um erro, pois não é possível usar uma função no CASE.

SELECT

CASE DATEPART(WEEKDAY, GETDATE())

WHEN 2

THEN 'Segunda-Feira'

WHEN 3

THEN 'Terça-Feira'

WHEN 4

THEN 'Quarta-Feira'

WHEN 5

THEN 'Quinta-Feira'

WHEN 6

THEN 'Sexta-Feira'

WHEN 7

THEN 'Sábado'

ELSE

'Domingo'

END AS NomeDiaSemana

(3) A função EOMONTH é utilizada para retornar o último dia do mês para a data que foi informada.

(4) A função ISDATE retorna 1 se a data informada for válida, caso contrário retorna -1.

a) V, F, V, V

b) V, F, V, F

c) F, V, F, V

d) F, F, V, V

e) F, F, V, F

# **(5.3) Mathematical Functions:**

**5)** Assinale a alternativa **CORRETA**:

(1) O resultado do SELECT abaixo será “124.00000”.

SELECT ROUND(123.99945, 2)

(2) O resultado do SELECT abaixo será “1, 0, 0”.

SELECT SIGN(2.4567), SIGN(0), SIGN(-2.4567)

(3) O comando abaixo sempre irá gerar números aleatórios.

SELECT RAND(100), RAND(), RAND(), RAND()

(4) O resultado do SELECT abaixo será “100” e “150”.

SELECT ABS(100), ABS(-150)

a) V, F, F, V

b) V, F, V, V

c) V, V, F, V

d) F, F, V, V

e) F, V, V, V

**6)** Assinale a alternativa **CORRETA**:

(1) O SQL Server não possui funções de trigonometria (seno, cosseno e tangente), pois é um banco de dados relacional.

(2) O resultado do comando abaixo será “2” e “1”.

SELECT FLOOR(1.99), CEILING(1.99)

(3) Na função POWER podemos especificar um número e escolher o valor da potência, enquanto a função SQUARE sempre irá elevar um número ao quadrado (potência de 2).

SELECT POWER(2, 3)

SELECT SQUARE(2)

(4) A função SQRT calcula a raiz quadrada do número informado.

SELECT SQRT(4)

a) F, F, V, F

b) F, V, V, V

c) F, F, V, V

d) V, F, F, V

e) V, F, V, V

# **(5.4) NULL Functions:**

**7)** Assinale a alternativa **CORRETA**:

(1) As funções ISNULL e COALESCE possuem a mesma sintaxe e fazem a mesma coisa.

(2) A função ISNULL pode receber vários parâmetros, enquanto a COALESCE recebe apenas dois.

(3) O comando abaixo irá retornar “Vitor”, pois é o segundo elemento da lista.

SELECT COALESCE(2, 'Luiz', 'Vitor', 'Lima')

(4) O comando abaixo irá retornar “100”.

DECLARE @VAR\_TESTE INT = 1

IF (@VAR\_TESTE <> NULL)

BEGIN

SELECT 100

END

ELSE

BEGIN

SELECT 200

END

GO

a) F, F, F, F

b) F, V, F, F

c) F, V, F, V

d) V, F, V, F

e) V, F, F, F

# **(5.5) String Functions:**

**8)** Assinale a alternativa **CORRETA**:

(1) Os scripts abaixo retornam o mesmo resultado:

-- SCRIPT 1

SELECT 'Luiz' + 'Vitor' + 'França' + 'Lima' + NULL

-- SCRIPT 2

SELECT CONCAT('Luiz','Vitor','França','Lima',NULL)

(2) O resultado do comando abaixo é “Luiz;Vitor;França;Lima”:

SELECT CONCAT\_WS(';', 'Luiz', NULL, 'Vitor', NULL, 'França', NULL, 'Lima')

(3) O resultado do comando abaixo é “LuizVitorFrançaLima”:

SELECT 'Luiz' + 'Vitor' + 'França' + 'Lima' + NULL

(4) O resultado do comando abaixo é “Luiz Vitor”:

DECLARE @Nome VARCHAR(50) = 'Luiz Vitor', @Sobrenome VARCHAR(50) = NULL

SELECT @Nome + ISNULL(' ' + @Sobrenome, '')

a) V, V, F, V

b) V, V, F, F

c) F, F, V, V

d) F, V, V, V

e) F, V, F, V

**9)** Assinale a alternativa **CORRETA**:

(1) O resultado do comando abaixo é “Luiz”:

SELECT SUBSTRING('Luiz Vitor', 4, 1)

(2) O resultado do comando abaixo é “Luiz” e “Vitor”:

SELECT LEFT('Luiz Vitor', 4), RIGHT ('Luiz Vitor', 5)

(3) O resultado do comando abaixo é “10” e “10”:

DECLARE @Nome VARCHAR(50) = 'Luiz ' -- 6 espaços em branco

SELECT LEN(@Nome), DATALENGTH(@Nome)

(4) O resultado do comando abaixo é “4” e “A”:

SELECT CHARINDEX('D', 'ABCDEFGHIJ')

SELECT PATINDEX('%[A-Z]%', '1234A678')

a) F, V, V, F

b) F, V, F, F

c) F, F, V, V

d) V, V, V, F

e) V, V, F, V

**10)** Assinale a alternativa **CORRETA**:

(1) O resultado do comando abaixo é “12345678974”:

SELECT REPLACE('123.456.789-74', '.', '')

(2) O resultado do comando abaixo é “00000”:

SELECT REPLICATE('0', 5)

(3) O resultado do comando abaixo é “luiz vitor” e “LUIZ VITOR”:

SELECT UPPER('LUIZ VITOR')

SELECT LOWER('luiz vitor')

(4) O resultado dos comandos abaixo será o mesmo “Luiz”:

SELECT RTRIM(LTRIM(' Luiz '))

SELECT TRIM(' Luiz ')

a) F, V, F, F

b) F, V, F, V

c) F, V, V, F

d) V, F, F, V

e) V, V, F, F

**11)** O comando abaixo retorna “xyabcz”.

SELECT STUFF('xyz', 2, 1, 'abc')

(V) Verdadeiro

(F) Falso

**12)** O comando abaixo retorna “ziuL rotiV”.

SELECT REVERSE('Luiz Vitor')

(V) Verdadeiro

(F) Falso