

# Desafio do Módulo 2

**Entrega** 5 ago em 19:30

**Pontos** 40

**Perguntas** 15

**Disponível** até 5 ago em 19:30

**Limite de tempo** Nenhum

## Instruções

O Desafio do Módulo 2 está disponível!

### 1. Instruções para realizar o desafio

Consulte a data de entrega no teste e em seu calendário.

Reserve um tempo para realizar a atividade, leia as orientações e enunciados com atenção. Em caso de dúvidas utilize o "Fórum de dúvidas do Desafio".

Para iniciá-lo clique em "Fazer teste". Você tem somente **uma** tentativa e não há limite de tempo definido para realizá-lo. Caso precise interromper a atividade, apenas deixe a página e, ao retornar, clique em "Retomar teste".

Clique em "Enviar teste" **somente** quando você concluí-lo. Antes de enviar confira todas as questões.

O gabarito será disponibilizado partir de sexta-feira, **07/08/2020**, às 21h.

Bons estudos!

### 2. O arquivo abaixo contém o enunciado do desafio

[Enunciado do Desafio - Módulo 2 - Bootcamp Banco de Dados.pdf](#) 

## Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	<a href="#">Tentativa 1</a>	1.177 minutos	40 de 40

! As respostas corretas estarão disponíveis em 7 ago em 21:00.

Pontuação deste teste: **40** de 40

Enviado 2 ago em 13:30

Esta tentativa levou 1.177 minutos.

### Pergunta 1

2,67 / 2,67 pts

Ao executar o script DDL abaixo, está sendo colocada uma restrição no schema físico, de forma a não deixar deletar um fornecedor que possui ao menos um serviço cadastrado no banco de dados.

```
ALTER TABLE SERVICO
```

```
ADD CONSTRAINT FK_FORNEC_SERVICO_01 FOREIGN  
KEY (COD_FORNECEDOR)
```

```
REFERENCES FORNECEDOR(COD_FORNECEDOR)
```

```
GO
```



Falso



Verdadeiro

### Pergunta 2

2,67 / 2,67 pts

Para gerar a lista única com os contatos (nome, endereço e telefone) de clientes e fornecedores, sem informações repetidas, ordenada alfabeticamente, foi construída a query abaixo:

```
SELECT DISTINCT NOM_CLIENTE AS "Nome da Pessoa",  
DSC_ENDERECO_COBRANCA AS "Endereço",
```

```
NUM_TELEFONE AS "Telefone"
```

```
FROM CLIENTE
```

UNION

```
SELECT DISTINCT NOM_FORNECEDOR AS "Nome da  
Pessoa", DSC_ENDERECO AS "Endereço", NUM_TELEFONE  
AS "Telefone"
```

```
FROM FORNECEDOR
```

```
ORDER BY "Nome da Pessoa" ASC
```

```
GO
```

**Nesta query, a opção DISTINCT pode ser suprimida, que a lista retornada será a mesma.**



Verdadeiro



Falso

### Pergunta 3

2,67 / 2,67 pts

Todos os comandos abaixo, permitirão a inserção da coluna IND\_STATUS na tabela FORMA\_PAGAMENTO, já populada, **EXCETO**:

```
ALTER TABLE FORMA_PAGAMENTO
```

```
ADD IND_STATUS char(4) NOT NULL DEFAULT 'NULL'
```



GO



```
ALTER TABLE FORMA_PAGAMENTO
```

```
ADD IND_STATUS char(10) NOT NULL DEFAULT 'NOT  
NULL'
```

```
GO
```

```
ALTER TABLE FORMA_PAGAMENTO
```

```
ADD IND_STATUS char(4) NOT NULL DEFAULT NULL
```



```
GO
```

```
ALTER TABLE FORMA_PAGAMENTO
```

```
ADD IND_STATUS char(1) NOT NULL DEFAULT 'D'
```



```
GO
```

#### Pergunta 4

2,67 / 2,67 pts

No comando abaixo, se forem encontrados valores nulos na coluna VLR\_TOTAL\_PEDIDO, o próprio script já insere um valor e realiza a alteração da coluna para NOT NULL.

```
IF EXISTS ( SELECT *
```

```
FROM PEDIDO
```

```
WHERE VLR_TOTAL_PEDIDO IS NULL
```

```
)
```

```
PRINT 'Existência de valores nulos para o campo'
```

```
--UPDATE PEDIDO
```

```
---SET VLR_TOTAL_PEDIDO=0
```

```
--WHERE VLR_TOTAL_PEDIDO IS NULL
```

```
ELSE
```

ALTER TABLE PEDIDO

ALTER COLUMN VLR\_TOTAL\_PEDIDO numeric (10,2) NOT  
NULL

GO

☐ Verdadeiro

☒ Falso

**Pergunta 5**

**2,67 / 2,67 pts**

O erro na query abaixo é devido à:

```
SELECT C.NOM_CLIENTE AS [NOME DO CLIENTE],  
P.NUM_PEDIDO PEDIDO, DAT_PEDIDO AS "DATA DO  
PEDIDO", P.VLR_TOTAL_PEDIDO  
  
FROM CLIENTE C JOIN PEDIDO P  
  
ON C.COD_CLIENTE = P.COD_CLIENTE  
  
WHERE YEAR ("DATA DO PEDIDO") = '2020'  
  
ORDER BY "NOME DO CLIENTE"  
  
GO
```

- 
- ☒ Alias de coluna sendo usado no filtro de dados
- 
- ☐ Falta de uso de alias de tabela na coluna DAT\_PEDIDO
- 
- ☐ Presença de espaço no alias da coluna NOM\_CLIENTE
- 
- ☐ Nenhuma das alternativas

### Pergunta 6

2,67 / 2,67 pts

Qual query retornará a lista de todos os serviços e os respectivos fornecedores, ordenados **alfabeticamente pelo nome do fornecedor** e **descendentemente pelo valor do serviço**?



```
SELECT F.NOM_FORNECEDOR, F.NUM_TELEFONE,  
S.DSC_SERVICO, S.TIP_SERVICO, S.VLR_SERVICO  
  
FROM SERVICO S  
  
JOIN FORNECEDOR F ON F.COD_FORNECEDOR =  
S.COD_FORNECEDOR  
  
ORDER BY F.NOM_FORNECEDOR ASC, S.VLR_SERVICO  
DESC  
  
GO
```



```
SELECT F.NOM_FORNECEDOR, F.NUM_TELEFONE,  
S.DSC_SERVICO, S.TIP_SERVICO, S.VLR_SERVICO  
  
FROM SERVICO S  
  
INNER JOIN FORNECEDOR F ON F.COD_FORNECEDOR =  
S.COD_FORNECEDOR  
  
ORDER BY F.NOM_FORNECEDOR ASC, S.VLR_SERVICO  
DESC  
  
GO
```



```
SELECT F.NOM_FORNECEDOR, F.NUM_TELEFONE,  
S.DSC_SERVICO, S.TIP_SERVICO, S.VLR_SERVICO  
  
FROM SERVICO S, FORNECEDOR F  
  
WHERE F.COD_FORNECEDOR = S.COD_FORNECEDOR  
  
ORDER BY F.NOM_FORNECEDOR ASC, S.VLR_SERVICO  
DESC  
  
GO
```



Todas as opções

**Pergunta 7**

**2,67 / 2,67 pts**

Com a query abaixo, é possível listar os serviços que nunca foram vendidos, ou que tiveram um total de vendas menor que 5 em 2019.

```
SELECT S.DSC_SERVICO, S.TIP_SERVICO,  
S.VLR_SERVICO, F.NOM_FORNECEDOR  
  
FROM SERVICO S, FORNECEDOR F  
  
WHERE S.COD_FORNECEDOR = F.COD_FORNECEDOR  
  
AND S.COD_SERVICO NOT IN (SELECT COD_SERVICO  
FROM ITEM_PEDIDO)  
  
UNION  
  
SELECT S.DSC_SERVICO, S.TIP_SERVICO,  
S.VLR_SERVICO, F.NOM_FORNECEDOR  
  
FROM SERVICO S, ITEM_PEDIDO I, FORNECEDOR F  
  
WHERE S.COD_SERVICO = I.COD_SERVICO  
  
AND S.COD_FORNECEDOR = F.COD_FORNECEDOR  
  
GROUP BY S.DSC_SERVICO, S.TIP_SERVICO,  
S.VLR_SERVICO, F.NOM_FORNECEDOR  
  
HAVING COUNT (I.COD_SERVICO) < 5  
  
GO
```



Verdadeiro



Falso

**Pergunta 8**

**2,67 / 2,67 pts**



A query abaixo pode ser usada para atualizar a coluna VLR\_TOTAL\_PEDIDO nos casos em que a coluna VLR\_TOTAL\_PEDIDO da tabela PEDIDO está nula ou com valor total calculado erroneamente.

```
UPDATE PEDIDO  
  
SET VLR_TOTAL_PEDIDO = P1.VLR_TOTAL_ITENS  
  
FROM PEDIDO P  
  
JOIN (SELECT P.NUM_PEDIDO, P.VLR_TOTAL_PEDIDO,  
SUM(I.VLR_TOTAL_ITEM) as VLR_TOTAL_ITENS  
  
FROM PEDIDO P JOIN ITEM_PEDIDO I  
  
ON P.NUM_PEDIDO = I.NUM_PEDIDO  
  
GROUP BY P.NUM_PEDIDO, P.VLR_TOTAL_PEDIDO  
  
HAVING P.VLR_TOTAL_PEDIDO !=  
SUM(I.VLR_TOTAL_ITEM) OR P.VLR_TOTAL_PEDIDO IS  
NULL) P1  
  
ON P.NUM_PEDIDO = P1.NUM_PEDIDO  
  
GO
```



Verdadeiro



Falso

### Pergunta 9

2,67 / 2,67 pts

Assinale a opção que listará corretamente o percentual de clientes cadastrados que estão com a coluna NUM\_CPF\_CNPJ nula.



```
SELECT CAST
(
    (      SELECT CAST(COUNT (*) AS numeric (3,2))
          FROM CLIENTE WHERE NUM_CPF_CNPJ IS
NULL
        )
    /
    (      SELECT CAST(COUNT (NUM_CPF_CNPJ)
AS numeric (3,2))
          FROM CLIENTE
        ) * 100 AS int
) AS "Percentual de Clientes sem Informação de Documento"
GO
```



```
SELECT CAST(
(SELECT CAST(COUNT (COALESCE (NUM_CPF_CNPJ,0))
AS numeric (3,2))
FROM CLIENTE WHERE NUM_CPF_CNPJ IS NULL
)
/
(SELECT CAST(COUNT (COALESCE (NUM_CPF_CNPJ,0))
AS numeric (3,2))
FROM CLIENTE
) * 100 AS int
) AS "Percentual de Clientes sem Informação de Documento"
GO
```



Todas as alternativas.



```
SELECT CAST
(
    (      SELECT CAST(COUNT (NUM_CPF_CNPJ)
AS numeric (3,2))
          FROM CLIENTE WHERE NUM_CPF_CNPJ IS
NULL
        )
    /
    (      SELECT CAST(COUNT (NUM_CPF_CNPJ)
AS numeric (3,2))
          FROM CLIENTE
        ) * 100 AS int
) AS "Percentual de Clientes sem Informação de Documento"
GO
```

### Pergunta 10

2,67 / 2,67 pts

Na query abaixo, que retorna a lista de clientes que não fizeram pedidos, pode-se substituir o NOT IN por NOT EXISTS ou por um LEFT JOIN.

```
SELECT NOM_CLIENTE, NUM_TELEFONE
FROM CLIENTE
WHERE COD_CLIENTE NOT IN (SELECT COD_CLIENTE
FROM PEDIDO)
ORDER BY NOM_CLIENTE
GO
```



Verdadeiro

☐ Falso

### Pergunta 11

2,67 / 2,67 pts

A query abaixo retornará o nome e o telefone dos clientes que já fizeram pedidos no ano corrente.

```
SELECT C.NOM_CLIENTE, C.NUM_TELEFONE
FROM CLIENTE C
WHERE EXISTS (
    SELECT P.COD_CLIENTE
    FROM PEDIDO P
    WHERE P.COD_CLIENTE =
C.COD_CLIENTE
    AND YEAR(DAT_PEDIDO) = YEAR
(GETDATE())
)
ORDER BY C.NOM_CLIENTE
GO
```

☒ Verdadeiro

☐ Falso

### Pergunta 12

2,67 / 2,67 pts

Assinale a opção que atende à solicitação da empresa de elaborar **uma query para gerar uma cópia dos dados da tabela ITEM\_PEDIDO** com o nome **TMP\_ITEM\_PEDIDO**.

☐ Todas as alternativas

BULK INSERT TMP\_ITEM\_PEDIDO  
FROM ITEM\_PEDIDO

☐ GO

CREATE TABLE TMP\_ITEM\_PEDIDO  
(  
    NUM\_PEDIDO int NOT NULL ,  
    COD\_ITEM\_PEDIDO int NOT NULL ,  
    COD\_SERVICO int NOT NULL ,  
    QTD\_SERVICO numeric(2) NOT NULL ,  
    VLR\_TOTAL\_ITEM numeric(10,2) NOT NULL  
)  
GO

INSERT INTO TMP\_ITEM\_PEDIDO  
SELECT \* FROM ITEM\_PEDIDO

☐ GO

SELECT \* INTO TMP\_ITEM\_PEDIDO  
FROM ITEM\_PEDIDO

☒ GO

**Pergunta 13**

**2,67 / 2,67 pts**

A query abaixo, ao ser executada, excluirá os clientes que estão cadastrados no banco de dados, mas que não fizeram nenhum pedido.

```
DELETE FROM CLIENTE  
  
FROM CLIENTE c  
  
LEFT JOIN PEDIDO p ON c.COD_CLIENTE =  
p.COD_CLIENTE  
  
WHERE p.COD_CLIENTE IS NULL  
  
GO
```

☒ Verdadeiro

☐ Falso

#### Pergunta 14

2,67 / 2,67 pts

Sobre a query abaixo, é **CORRETO** afirmar.

```
INSERT INTO FORMA_PAGAMENTO  
(COD_FORMA_PAGAMENTO, DSC_FORMA_PAGAMENTO)  
VALUES(799,'MERCADO PAGO')  
  
GO
```

☐ Insere dados na tabela FORMA\_PAGAMENTO, independentemente da ordem em que as colunas da tabela foram criadas.



Se a tabela possuir a coluna PERCENTUAL\_TAXA int NULL, o comando será executado com sucesso, caso não ocorra duplicidade de dados.



Se a tabela possuir a coluna IND\_STATUS char(1) NOT NULL, e esta coluna não possuir uma constraint DEFAULT, o comando falhará.



Todas as alternativas

### Pergunta 15

2,62 / 2,62 pts

Acerca da execução do bloco SQL abaixo, é **INCORRETO** afirmar:

```
BEGIN TRANSACTION Transacao1
```

```
INSERT INTO FORMA_PAGAMENTO  
(COD_FORMA_PAGAMENTO,  
DSC_FORMA_PAGAMENTO)  
VALUES (199, 'CARTÃO DE DÉBITO');
```

```
BEGIN TRANSACTION Transacao2
```

```
INSERT INTO FORMA_PAGAMENTO  
(COD_FORMA_PAGAMENTO,  
DSC_FORMA_PAGAMENTO)  
VALUES (299, 'CARTÃO DE CRÉDITO');
```

COMMIT TRANSACTION Transacao2

INSERT INTO FORMA\_PAGAMENTO

(COD\_FORMA\_PAGAMENTO,  
DSC\_FORMA\_PAGAMENTO)

VALUES (399, 'BOLETO');

ROLLBACK

COMMIT TRANSACTION Transacao1

INSERT INTO FORMA\_PAGAMENTO

(COD\_FORMA\_PAGAMENTO, DSC\_FORMA\_PAGAMENTO)

VALUES (499, 'FATURA');



Serão inseridas 4 linhas na tabela.



Serão abertas 3 transações, sejam elas explícitas ou implícitas.



O commit da transação Transacao1 falhará.



Será inserida apenas 1 linha na tabela.

Pontuação do teste: **40** de 40