Q1 – FCC – MPE-PB - 2015

Atenção: Considere as informações abaixo para responder à questão.

```
CREATE TABLE Loja (
    idLoja INTEGER NOT NULL,
    nomeLoja VARCHAR(45),
    PRIMARY KEY(idLoja)
);
CREATE TABLE Filial (
    idFilial INTEGER NOT NULL,
    idLoja INTEGER NOT NULL,
    cidadeFilial VARCHAR(45),
    vendasFilial DOUBLE,
     PRIMARY KEY(idFilial, idLoja),
        FOREIGN KEY(idLoja)
        REFERENCES Loja(idLoja)
);
```

Considere que na tabela Loja foram cadastrados os seguintes registros:

idLoja	nomeLoja
1	A
2	B
3	C

Na tabela Filial, foram cadastrados os seguintes registros:

idFilial	idLoja	cidadeFilial	vendasFilial
1	1	Campinas	120000
1	2	Campinas	35000
1	3	São Paulo	120000
2	1	Salvador	240000

2	2	Salvador	20000
3	1	Rio de Janeiro	60000
4	1	Curitiba	40000

Ao tentar executar a instrução INSERT INTO filial (idFilial, idLoja, cidadeFilial, vendasFilial) VALUES (2, 1, 'São Paulo', 340000); será informado pelo SGBD que

a) há um erro no nome da cidade, pois deveria estar entre aspas duplas.

b) não foi possível adicionar o registro porque o id da loja não está cadastrado.

c) há um erro na sintaxe da instrução SQL.

d) o script SQL foi aplicado com sucesso no banco de dados.

e) houve uma tentativa de entrada duplicada '2-1' para a chave primária.

Q2 – IDECAN – INMETRO - 2015

Deve-se criar uma sintaxe, em SQL, para popular uma tabela de cadastro de funcionários. O nome da tabela é Cadastro e tem o seguinte formato:

CODFUN	NOME	DEPTO	FUNCAO	SALÁRIO

Observe a tabela com um registro inserido.

CODFUN	NOME	DEPTO	FUNCAO	SALÁRIO
11	Fulano de Tal	05	Balconista	1200.00

Assinale a sintaxe correta para inserção desse registro na tabela Cadastro.

a) INSERT INTO Cadastro VALUES (11, 'Fulano de tal', '05', 'Balconista', 1.200,00).

b) INSERT INTO Cadastro VALUES (CODFUN, NOME, DEPTO, FUNCAO, SALARIO; 11, 'Fulano de tal', '05', 'Balconista', 1200.00).

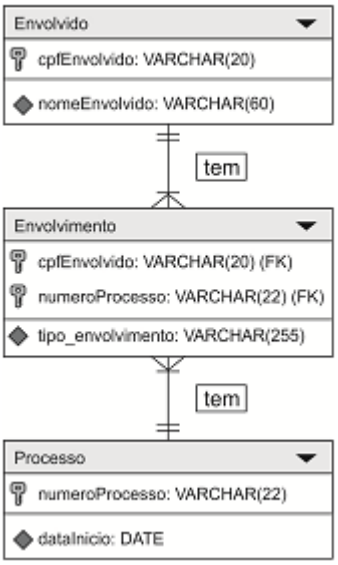
c) INSERT INTO Cadastro (CODFUN, NOME, DEPTO, FUNCAO, SALARIO) VALUES (11, 'Fulano de tal', '05', 'Balconista', 1200.00).

d) INSERT INTO Cadastro (CODFUN, NOME, DEPTO, FUNCAO, SALARIO) VALUES (11, Fulano de tal, 05, Balconista, 1200.00).

e) INSERT INTO Cadastro (CODFUN, NOME, DEPTO, FUNCAO, SALARIO) VALUES (11, 'Fulano de tal', '05', 'Balconista', 1200,00).

Q3 - FCC – MANAUSPREV – An. Previdênciário – 2015

Modelo Entidade-Relacionamento



Dados cadastrados nas tabelas:

cpfEnvolvido	nomeEnvolvido
121.134.045-01	Marcos Paulo
128.249.039-14	Maria de Fátima
131.091.431-09	André Luiz
158.245.067-12	Pedro nda Silva
160.234.074-11	João da Silva

Observação: O erro no nome `Pedro nda Silva` é proposital.

cpfEnvolvido	numeroProcesso	tipo_envolvimento
121.134.045-01	12345670020108261234	acusado
128.249.039-14	12345670020108261234	acusador
128.249.039-14	78654310020118150675	acusado
131.091.431-09	16789000020105131234	acusado
158.245.067-12	73982110120111131001	acusado
158.245.067-12	78654310020118150675	acusador
158.245.067-12	98006770120111144571	acusado

numeroProcesso	dataInicio
12345670020108261234	20/01/2015
16789000020105131234	08/08/2010
73982110120111131001	10/12/2014
78654310020118150675	30/01/2015
82745660320095202300	22/06/2014
98006770120111144571	21/02/2015

O Administrador de Banco de Dados digitou um conjunto de instruções no banco de dados aberto e em condições ideais, após as tabelas terem sido criadas e os dados terem sido cadastrados. Considere que nenhuma operação, além da criação das tabelas e do cadastramento, havia sido realizada. Dentre as instruções digitadas, a única instrução que executa corretamente é

a) INSERT INTO Processo VALUES ('1234567.00.2010.8.26.1234', '2015-01-20');

b) DELETE FROM Processo WHERE numeroProcesso='73982110120111131001';

c) ALTER TABLE Envolvimento DROP COLUMN numeroProcesso;

d) INSERT INTO Envolvimento (cpfEnvolvido, numeroProcesso, tipo_envolvimento) VALUES ('158.245.067-12', '58654310020254589031', 'acusado');

e) UPDATE Envolvido SET nomeEnvolvido='Pedro da Silva' WHERE cpfEnvolvido='158.245.067-12';

Q4 – FCC – MPE-PB - 2015

Tabela **Proc_Cidades**

Nome_Cidade	Data_Proc	Processos
João Pessoa	05-Jan-2014	1500
Campina Grande	07-Jan-2014	250
João Pessoa	08-Jan-2014	300
Santa Rita	08-Jan-2014	700

Tabela **Proc_Datas**

Data_Proc	Processos
07-Jan-2014	250
10-Jan-2014	535
11-Jan-2014	320
12-Jan-2014	750

Tabela **Geografia**

Regiao	Nome_Cidade
Mata	João Pessoa
Agreste	Campina Grande
Borborema	Monteiro
Mata	Santa Rita

A instrução SQL capaz de listar todas as diferentes datas em que foram registrados processos é:

a) SELECT Data_Proc FROM Proc_Cidades UNION SELECT Data_Proc FROM Proc_Datas;

b) SELECT Data_Proc FROM Proc_Cidades HAVING Data_Proc FROM Proc_Datas;

c) SELECT Data_Proc FROM Proc_Cidades AND * FROM Proc_Datas;

d) SELECT Data_Proc FROM Proc_Datas GROUP BY Data_Proc FROM Proc_Cidades;

e) SELECT Data_Proc FROM Proc_Cidades UNION ALL SELECT Proc_Datas FROM Data_Proc;

Q5 – FCC – TER-RR - 2015

Considere a instrução SQL a seguir:

SELECT Clientes.NomeCliente, Pedidos.PedidoID FROM Clientes

...I....

ON Clientes.ClienteID=Pedidos.ClienteID

ORDER BY Clientes.NomeCliente;

Esta instrução seleciona todas as linhas de ambas as tabelas, desde que haja uma correspondência entre as colunas ClienteID. Se houver linhas na tabela Clientes que não tem correspondentes na tabela Pedidos, esses clientes não serão listados.

Para que a instrução dê o resultado descrito, a lacuna I deve ser preenchida com

a) INNER JOIN Pedidos

b) LEFT JOIN

c) RIGHT OUTER JOIN

d) FULL OUTER JOIN

e) LEFT OUTER JOIN

Q6 – FGV – TCE-SE - 2015

X		Y	
a	b	c	d
1	2	1	2
3	3	3	4
4	5	5	6
5	7	7	8
		9	1

select *

from x left join y on x.a=y.c

order by x.a

Considerando-se as tabelas e o comando SQL mostrados acima, é correto concluir que esse comando produz:

a)

1, 2, 1, 2

3, 3, 3, 4

5, 7, 5, 6

b)

1, 2, 1, 2

3, 4, 3, 3

NULL, NULL, 4, 5

5, 6, 5, 7

c)

1, 2, 1, 2

3, 3, 3, 4

NULL, NULL, NULL, NULL

5, 7, 5, 6

d)

1, 2, 1, 2

3, 3, 3, 4

4, 5, NULL, NULL

5, 7, 5, 6

e)

1, 2

3, 3

4, 5

5, 7

Q7 – FUNDATEC – BRDE - 2015

Para responder à questão, considere a Figura 1 e a Figura 2.

id	name	counter1	counter2
1	Any	10	8
2	Peter	5	7
3	Alice	12	0
4	Bob	9	9
5	Gil	NULL	NULL

Figura 1 – Tabela users

id	owner	label	value
1	1	martelo	20
2	1	relógio	130
3	3	carro	24000
4	4	tv	1200
5	5	livro	49

Figura 2 – Tabela stuff

Considerando a tabela users mostrada na Figura 1 e a tabela stuff apresentada na Figura 2, qual o número de registros retornado pela consulta SQL abaixo?

SELECT * FROM users LEFT JOIN stuff on users.id = stuff.owner AND stuff.value > 1000;

a) 0

b) 2

c) 5

d) 6

e) 25

Q8 – CESPE – MEC - 2015

A manipulação dos dados armazenados em bancos de dados é realizada, muitas vezes, por linguagens com essa finalidade específica. Essas linguagens podem ser próprias do SGBD utilizado, embora muitas vezes representem a implementação de linguagem padrão de acesso a dados. Acerca desse assunto, julgue o seguinte item.

- Na linguagem SQL, o comando FULL OUTER JOIN combina os resultados dos comandos LEFT JOIN e RIGHT JOIN.

Q9 – FCC – MPE-PB - 2015

Tabela **Proc_Cidades**

Nome_Cidade	Data_Proc	Processos
João Pessoa	05-Jan-2014	1500
Campina Grande	07-Jan-2014	250
João Pessoa	08-Jan-2014	300
Santa Rita	08-Jan-2014	700

Tabela **Proc_Datas**

Data_Proc	Processos
07-Jan-2014	250
10-Jan-2014	535
11-Jan-2014	320
12-Jan-2014	750

Tabela **Geografia**

Regiao	Nome_Cidade
Mata	João Pessoa
Agreste	Campina Grande
Borborema	Monteiro
Mata	Santa Rita

A instrução SQL a seguir:

```
SELECT SUM(Processos) FROM Proc_Cidades WHERE EXISTS (SELECT * FROM Geografia
WHERE Regiao = 'Mata');
```

resulta em

a) 1800.

b) 2050.

c) Não retorna nada.

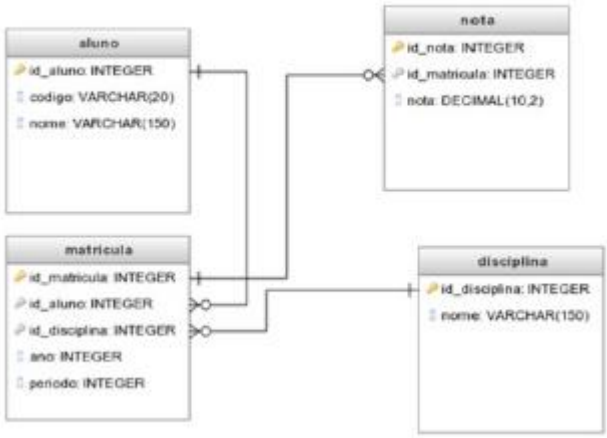
d) 2750.

e) 2500.

Q10 – UFCG – UFCG - 2016

Marque a alternativa que contenha a consulta correta para retornar o nome dos alunos que não estão matriculados em disciplinas.

Considere o uso de SQL:2011 / ISO/IEC 9075:2011.



a)

SELECT aluno.nome FROM aluno INNER JOIN matricula ON matricula.id_aluno = aluno.id_aluno WHERE aluno.id_aluno IS NULL;

b)

SELECT aluno.nome FROM aluno RIGHT OUTER JOIN matricula ON matricula.id_aluno = aluno.id_aluno WHERE matricula.id_aluno IS NULL;

c)

SELECT aluno.nome FROM aluno WHERE NOT IN (SELECT * FROM matricula WHERE matricula.id_aluno = aluno.id_aluno);

d)

SELECT aluno.nome FROM aluno WHERE NOT EXISTS (SELECT * FROM matricula WHERE matricula.id_aluno = aluno.id_aluno);

e)

SELECT aluno.nome FROM aluno WHERE EXISTS (SELECT * FROM matricula WHERE matricula.id_aluno != aluno.id_aluno);

G**abarito**

Q1 – E

Q2 – C

Q3 – E

Q4 – A

Q5 – A

Q6 – D

Q7 – C

Q8 – C

Q9 – D

Q10 – D