|  |  |  |  |
| --- | --- | --- | --- |
| Desenho de personagem de desenhos animados com texto preto sobre fundo branco  Descrição gerada automaticamente com confiança média | Ano Letivo: 20251 | Turno: Noturno | Local:  A definir |
| Curso:  ENGENHARIA DE SOFTWARE/COMPUTAÇÃO | | Turma: 3ª Fase |
| Disciplina:  BANCO DE DADOS II | | Professor(a):  JORGE LUIZ DA SILVA | |

Revisão Programação T-SQL, Stored Procedure, Trigger, Function

Para responder os exercícios abaixo utilize o padrão de Linguagem do SGBD Microsoft SQL Server.

Você poderá criar um banco de dados SQL no Azure, através de uma assinatura MS LEARN SANDBOX e criar os dados e tabelas através do arquivo **script\_2022.txt** (anexo ao AVA). Link abaixo:

[https://learn.microsoft.com/pt-br/training/modules/build-serverless-api-with-functions-api-management/5-](https://learn.microsoft.com/pt-br/training/modules/build-serverless-api-with-functions-api-management/5-exercise-import-additional-functions-existing-api-gateway?ns-enrollment-type=learningpath&ns-enrollment-id=learn.create-serverless-applications) [exercise-import-additional-functions-existing-api-gateway?ns-enrollment-type=learningpath&ns-enrollment-](https://learn.microsoft.com/pt-br/training/modules/build-serverless-api-with-functions-api-management/5-exercise-import-additional-functions-existing-api-gateway?ns-enrollment-type=learningpath&ns-enrollment-id=learn.create-serverless-applications) [id=learn.create-serverless-applications](https://learn.microsoft.com/pt-br/training/modules/build-serverless-api-with-functions-api-management/5-exercise-import-additional-functions-existing-api-gateway?ns-enrollment-type=learningpath&ns-enrollment-id=learn.create-serverless-applications)

Ou usar a instância de SQL Server SQLSDEV instalada em seu computador.

# (múltipla escolha) Uma store procedure suporta:

* 1. Nenhum parâmetro.
  2. Somente parâmetros de entrada.
  3. Parâmetros de entrada e saída.
  4. Somente parâmetros de saída.

# (múltipla escolha) Uma trigger suporta:

* 1. Nenhum parâmetro.
  2. Somente parâmetros de entrada.
  3. Parâmetros de entrada e saída.
  4. Somente parâmetros de saída.

# (múltipla escolha) Uma função suporta:

* 1. Nenhum parâmetro.
  2. Somente parâmetros de entrada.
  3. Parâmetros de entrada e saída.
  4. Somente parâmetros de saída.

# Considere VERDADEIRO ou FALSO para as afirmativas abaixo:

(F) Trigger são criadas por CREATE TRIGGER e executadas por EXECUTE. (V) Funções são criadas por CREATE FUNCTION e executadas por SELECT.

(V) Stored procedures são criadas por CREATE PROCEDURE e executados por EXECUTE.

(V) Triggers são executadas automaticamente conforme o disparo de um evento no banco de dados.

(F) Funções são executadas automaticamente por um evento no banco de dados.

(F) Stored procedures são executadas automaticamente pelo disparo de um evento no banco de dados.

(F) Funções são criadas por CREATE FUNCTION e executadas por EXECUTE.

# (objetiva) A diferença básica entre uma trigger, uma stored procedure e uma function é que, respectivamente:

* 1. Todas são executadas de acordo com um evento, mas não ficam dentro de um banco de dados.
  2. É executada de acordo com um evento; é chamada para ser executada; é chamada para ser executada e todas elas ficam dentro de um banco de dados.
  3. Todas são executadas após serem chamadas, porém a primeira não fica dentro de um banco de dados.
  4. Todas são executadas após serem chamadas, porém a segunda não fica dentro de um banco de dados.
  5. Todas são executadas após serem chamadas, porém a terceira não fica dentro de um banco de dados.
  6. É chamada para ser executada; É chamada para ser executada; é executada de acordo com um evento e todas elas não ficam dentro de um banco de dados.

# (múltipla escolha) Se considerarmos a assinatura da stored procedure a seguir, podemos afirmar que:

procedure x (@y int, @z int output)

* 1. A procedure retorna um valor inteiro após sua execução, através de uma variável.
  2. A procedure não recebe parâmetros de entrada.
  3. A procedure possui dois parâmetros de entrada.
  4. O nome da procedure é x e ela não retorna nenhum valor.
  5. O nome da procedure é x; y e z são parâmetros.
  6. y e z são os valores retornados pela procedure após sua execução.

# (múltipla escolha) Se considerarmos a assinatura da função escalar a seguir, podemos afirmar que:

function x (@y int) returns int

* 1. A função retorna um valor inteiro após sua execução.
  2. A função não recebe parâmetros de entrada.
  3. A função possui um parâmetro.
  4. O nome da função é x e ela não retorna nenhum valor.
  5. O nome da função é x; y é um parâmetro; e retorna um valor inteiro.

# (objetiva) A diferença básica entre uma trigger, uma stored procedure e uma function é que, respectivamente:

* 1. Todas são executadas de acordo com um evento, mas não ficam dentro de um banco de dados.
  2. É executada de acordo com um evento; é chamada para ser executada; é chamada para ser executada e todas elas ficam dentro de um banco de dados.
  3. Todas são executadas após serem chamadas, porém a primeira não fica dentro de um banco de dados.
  4. Todas são executadas após serem chamadas, porém a segunda não fica dentro de um banco de dados.
  5. Todas são executadas após serem chamadas, porém a terceira não fica dentro de um banco de dados.
  6. É chamada para ser executada; É chamada para ser executada; é executada de acordo com um evento e todas elas não ficam dentro de um banco de dados.

# (múltipla escolha) Se considerarmos a assinatura da stored procedure a seguir, podemos afirmar que:

procedure x (@y int, @z int output)

* 1. A procedure retorna um valor inteiro após sua execução, através de uma variável.
  2. A procedure não recebe parâmetros de entrada.
  3. A procedure possui dois parâmetros de entrada.
  4. O nome da procedure é x e ela não retorna nenhum valor.
  5. O nome da procedure é x; y e z são parâmetros.
  6. y e z são os valores retornados pela procedure após sua execução.

# (múltipla escolha) Se considerarmos a assinatura da função escalar a seguir, podemos afirmar que:

function x (@y int) returns int

* 1. A função retorna um valor inteiro após sua execução.
  2. A função não recebe parâmetros de entrada.
  3. A função possui um parâmetro.
  4. O nome da função é x e ela não retorna nenhum valor.
  5. O nome da função é x; y é um parâmetro; e retorna um valor inteiro.

# Sobre os temas de procedimentos armazenados (stored procedures) e gatilhos (triggers) e funções (functions), marque somente as afirmações FALSAS.

(F) Uma trigger é equivalente a uma função, inclusive pelo fato de retornar um valor.

(V) A execução de uma stored procedure é feita por meio de chamada ao seu nome, podendo ser, inclusive,

(V) Triggers são executadas automaticamente conforme o disparo de um evento no banco de dados.

(F) Funções são executadas automaticamente por um evento no banco de dados.

(F) Stored procedures são executadas automaticamente pelo disparo de um evento no banco de dados.

(F) Funções são criadas por CREATE FUNCTION e executadas por EXECUTE.

# (objetiva) A diferença básica entre uma trigger, uma stored procedure e uma function é que, respectivamente:

* 1. Todas são executadas de acordo com um evento, mas não ficam dentro de um banco de dados.
  2. É executada de acordo com um evento; é chamada para ser executada; é chamada para ser executada e todas elas ficam dentro de um banco de dados.
  3. Todas são executadas após serem chamadas, porém a primeira não fica dentro de um banco de dados.
  4. Todas são executadas após serem chamadas, porém a segunda não fica dentro de um banco de dados.
  5. Todas são executadas após serem chamadas, porém a terceira não fica dentro de um banco de dados.
  6. É chamada para ser executada; É chamada para ser executada; é executada de acordo com um evento e todas elas não ficam dentro de um banco de dados.

# (múltipla escolha) Se considerarmos a assinatura da stored procedure a seguir, podemos afirmar que:

procedure x (@y int, @z int output)

* 1. A procedure retorna um valor inteiro após sua execução, através de uma variável.
  2. A procedure não recebe parâmetros de entrada.
  3. A procedure possui dois parâmetros de entrada.
  4. O nome da procedure é x e ela não retorna nenhum valor.
  5. O nome da procedure é x; y e z são parâmetros.
  6. y e z são os valores retornados pela procedure após sua execução.

# (múltipla escolha) Se considerarmos a assinatura da função escalar a seguir, podemos afirmar que:

function x (@y int) returns int

* 1. A função retorna um valor inteiro após sua execução.
  2. A função não recebe parâmetros de entrada.
  3. A função possui um parâmetro.
  4. O nome da função é x e ela não retorna nenhum valor.
  5. O nome da função é x; y é um parâmetro; e retorna um valor inteiro.

# Sobre os temas de procedimentos armazenados (stored procedures) e gatilhos (triggers) e funções (functions), marque somente as afirmações FALSAS.

(F) Uma trigger é equivalente a uma função, inclusive pelo fato de retornar um valor.

(V) A execução de uma stored procedure é feita por meio de chamada ao seu nome, podendo ser, inclusive, dentro de outra stored procedure.

(V) Uma trigger é uma stored procedure que é ativada por evento e executa uma ou mais ações

(F) Transações (commit e rollback) não podem ser executadas em stored procedures e funções, somente em triggers.

(V) O uso de stored procedure diminui o tráfego na rede em uma arquitetura client/server, pois os comandos SQL ficam armazenados em um banco de dados.

# Fulano monitora um servidor SQL Server e precisa criar algo que guarde todas as alterações ocorridas na estrutura das tabelas, especialmente os comandos de DDL (CREATE, ALTER e DROP).

**A estratégia mais apropriada para detectar comandos dessa natureza, considera a criação e uso de:**

* 1. Funções
  2. Identity
  3. Bloqueios
  4. Stored Procedures
  5. Triggers

# Você recebeu um ticket de suporte para avaliar o código abaixo e...

DECLARE @placa int

SET @placa = 'CCR8096' UPDATE carro

SET cor = 'BRANCO'

WHERE chassi IS\_NULL

AND ano BETWEEN 2020 and 2022

AND placa = @placa

PRINT 'O número de linhas atualizadas é : ' + CONVERT(varchar(100), @@ROWCOUNT)

**...concluiu corretamente que existem os seguintes erros na instrução (considere todos os erros encontrados).**

* 1. @@ROWCOUNT, pois deveria ser escrito com somente um caracter @
  2. IS\_NULL , deveria ser escrito dessa forma IS NULL
  3. CONVERT, por ser um comando completamente desnecessário nessa instrução
  4. PRINT, pois para mostrar a informação em tela precisaria ser utilizada a instrução ECHO
  5. BETWEEN 2020 AND 2022, pois os valores 2020 AND 2022 deveriam ser inseridos entre parênteses
  6. DECLARE @placa int, pois o tipo de dado correto deveria ser alfanumérico tipo varchar.
  7. SET @placa, pois o correto seria setar o valor da variável através do comando SELECT.

# Considere o código abaixo:

create proc x (@m int, @r datetime output) as begin

select @r = dateadd(MM, @m, getdate())

end go

**Responda qual é a finalidade desta stored procedure (considerar somente uma correta – a mais completa).**

* 1. Listar a data atual.
  2. Listar o mês atual.
  3. Informar o mês atual através da variável @r
  4. Informar o mês atual através da variável @m
  5. Informar uma data (com base na data atual), acrescida ou diminuída de @m mês(es), através de @r.
  6. Informar uma data, acrescida de @m mês(es), através de @r.
  7. Informar o mês (com base na data atual), acrescida de @m minutos, através de @r.

# Encontre o(s) erro(s) no código abaixo, efetue a(s) correção(ões) e poste o código corrigido:

create function fn\_count\_apolices (@cod\_cliente int) returns int

as begin

declare @resultado int, @result varchar(10)

select @resultado = count(cod\_apolice) from apolice where cod\_cliente = @cod\_cliente return @result

end go

Resposta:

create function fn\_count\_apolices (@cod\_cliente int) returns int

as begin

declare @resultado int

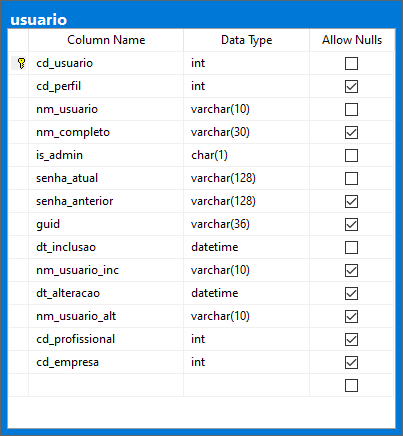
select @resultado = count(cod\_apolice) from apolice where cod\_cliente = @cod\_cliente return @resultado

end go

select dbo.fn\_count\_apolices(cod\_cliente) as qtde

from cliente

# Explique qual a finalidade da trigger abaixo:

****

create trigger tu\_usuario on usuario for update as begin

if rowcount\_big() = 0 return

declare @senha\_atual varchar(30), @cd\_usuario int

select @cd\_usuario = cd\_usuario, @senha\_atual = senha\_atual from inserted

end go

if update(senha\_atual) begin

update usuario

set senha\_anterior = @senha\_atual where cd\_usuario = @cd\_usuario

end

Resposta: A trigger é usada para alterar a senha\_anterior = senha atual, isso quando a senha atual for alterada, nesse caso, o evento de UPDATE aciona a trigger.

# Crie uma stored procedure que informe quantos carros possui cada cliente (lista). Escreva também os comandos usados na execução da stored procedure.

Resposta: create proc contar as

Begin

Select cliente.nome, count(carro.placa)

From cliente

Inner join apolice

On cliente.cod\_cliente = apolice.cod\_cliente

Inner join carro

On apolice.placa = carro.placa

Group by cliente.nome

End

Go

Exec contar

# Crie uma função que informe quantos carros possui cada cliente. Escreva também o comando usado na chamada da função (lista).

Resposta: create function contar (@cod\_cliente int) returns int as

Begin

Declare @resultado int

Select @resultado = count(apolice.placa)

From cliente

Inner join apolice

On cliente.cod\_cliente = apolice.cod\_cliente

Where cliente.cod\_cliente = @cod\_cliente

Return @resultado

End

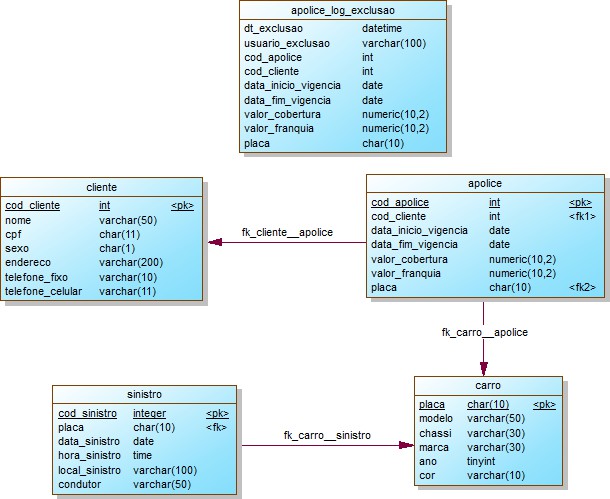
Go

Selec dbo.contar (cod\_cliente), nome

From cliente

# Dado o modelo abaixo, crie uma trigger que guarde todos os eventos de DELETE da tabela apolice, na tabela apolice\_log\_exclusao (você terá que criar a tabela apolice\_log\_exclusao). Use a função suser\_name() para capturar o nome do usuário e a getdate() para pegar a data e hora atual.

**Importante: Você deve contemplar exclusões massivas (mais de uma linha por evento) em seu código na trigger.**

****

Resposta: create table apolice\_log\_exclusao (

Dt\_exclusao datetime NULL,

Usuario\_exclusao varchar(100) NULL,

Cod\_apolice int NULL,

Cod\_cliente int NULL,

Data\_inicio\_vigencia date NULL,

Data\_fim\_vigencia date NULL,

Valor\_cobertura numeric(10,2) NULL,

Valor\_franquia numeric(10,2) NULL,

Placa char(10) NULL

)

go

Create trigger td\_excluir for delete as

Begin

If(rowcount\_big() = 0)

Return

Else

begin

Insert into apolice\_log\_exclusao

Select getdate(), suser\_name(), cod\_apolice, cod\_cliente, Data\_inicio\_vigencia, Data\_fim\_vigencia, Valor\_cobertura, Valor \_franquia, Placa

From deleted

End

End

go

# Crie uma função que retorne o número de apolices vinculadas a um cliente (usar campo código do cliente como parâmetro da função). Depois de criada a função, evidencie seu pleno funcionamento escrevendo um comando de select na tabela cliente, junto com a função criada.

Resposta:

# Crie uma stored procedure que retorne o número de apólices, vinculadas a um cliente, através de uma variável de saída (usar campo código do cliente como parâmetro de entrada). Depois de criada a stored procedure, evidencie seu pleno funcionamento escrevendo o comando de exemplo que execute e mostre o resultado da stored procedure.

Resposta: