TRIGGERS e STORED PROCEDURES.

```
================================================================
===
-- 1. Banco de Dados para Trigger de Atualização de Salários (Exercício 1)
--
================================================================
==

CREATE DATABASE Empresa;

USE Empresa;

CREATE TABLE Funcionarios(

     IDfuncionario INT AUTO_INCREMENT PRIMARY KEY ,

     nome VARCHAR(100),

     salario DECIMAL(10,2)

);


CREATE TABLE Historico_Salarios(

     IDhistorico INT AUTO_INCREMENT PRIMARY KEY,

     funcionarioID INT,

     salario_antigo DECIMAL(10,2),

     salario_novo DECIMAL(10,2),

     data_alteracao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

     FOREIGN KEY (funcionarioID) REFERENCES
Funcionarios(IDfuncionario)

);

-- Dados

INSERT INTO Funcionarios (IDfuncionario,nome, salario)

VALUES
```

```
(1,'Mario Silva', 2500.00),
(2,'Maria Oliveira', 3000.00),
(3,'Carlos Pereira', 2800.50),
(4,'Ana Santos', 3200.75);

INSERT INTO Historico_Salarios (funcionarioID,
salario_antigo, salario_novo)
VALUES
(1, 2400.00, 3500.00),
(2, 2900.00, 5000.00),
(3, 2700.00, 3800.50),
(4, 3100.00, 5200.75);
-- ========================================================
-- 2. Banco de Dados para Inserção de Produtos (Exercício2)
-- ========================================================
CREATE DATABASE Loja;
USE Loja;
CREATE TABLE Produtos(
     IDprodutos INT AUTO_INCREMENT PRIMARY KEY,
     nome VARCHAR(100),
     preco DECIMAL(10,2),
     categoria VARCHAR(50)
);

INSERT INTO Produtos (nome, preco, categoria)
VALUES
```

```
('Camiseta', 29.90, 'Roupas'),
('Calça Jeans', 89.90, 'Roupas'),
('Tênis Esportivo', 199.99, 'Calçados'),
('Relógio de Pulso', 299.50, 'Acessórios'),
('Notebook', 3499.00, 'Eletrônicos'),
('Smartphone', 1999.99, 'Eletrônicos'),
('Fone de Ouvido', 149.90, 'Acessórios'),
('Cadeira Gamer', 799.00, 'Mobiliário'),
('Mesa de Jantar', 1299.00, 'Mobiliário'),
('Cerveja Artesanal', 15.00, 'Bebidas');
-- ========================================================
-- 3. Banco de Dados para Registro de Exclusões de Clientes (Exercício 3)
-- ========================================================
CREATE DATABASE Vendas_t;
USE Vendas_t;
CREATE TABLE Clientes (
     IDcliente INT AUTO_INCREMENT PRIMARY KEY,
     nome VARCHAR(100),
     email VARCHAR(100),
     telefone VARCHAR(15)
);

CREATE TABLE Clientes_Excluidos(
     IDexcluidos INT AUTO_INCREMENT PRIMARY KEY,
     nome VARCHAR(100),
     email VARCHAR(100),
```

```
    telefone VARCHAR(15),
    data_exclusao TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- dados
INSERT INTO Clientes (nome, email, telefone)
VALUES
('Mario Silva', 'joao.silva@email.com', '11987654321'),
('Mariana Oliveira', 'maria.oliveira@email.com',
'21987654321'),
('Alberto Pereira', 'carlos.pereira@email.com',
'31987654321'),
('Ana Flavia Santos', 'ana.santos@email.com',
'41987654321'),
('Luiz Joaquim Fernando', 'luiz.fernando@email.com',
'51987654321');



-- ======================================================
-- 4. Banco de Dados para Cálculo de Média (Exercício 4)
-- ======================================================
CREATE DATABASE Escola_t;
USE Escola_t;

CREATE TABLE Alunos(
    IDalunos INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100)
```

```
);

CREATE TABLE Notas(
     IDnotas INT AUTO_INCREMENT PRIMARY KEY,
     alunosID INT,
     nota DECIMAL(10,2),
     FOREIGN KEY (alunosID) REFERENCES Alunos(IDalunos)

);
-- dados
INSERT INTO Alunos (nome)
VALUES
('Alice Silva'),
('Bruno Santos'),
('Carlos Oliveira'),
('Diana Costa');

INSERT INTO Notas (alunosID, nota)
VALUES
(1, 8.5),
(2, 7.0),
(1, 9.0),
(3, 6.5),
(4, 10.0);
```

```
-- ========================================================
-- 5. ESTOQUE
-- ========================================================
CREATE DATABASE Estoque_t;
USE Estoque_t;

CREATE TABLE Estoque(
     IDestoque INT AUTO_INCREMENT PRIMARY KEY,
     Nome VARCHAR(100),
     quantidade INT
);

CREATE TABLE Vendas(
     IDvendas INT AUTO_INCREMENT PRIMARY KEY,
     produto_estoqueID INT,
     quantidade INT,
     data_venda TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
     FOREIGN KEY (produto_estoqueID) REFERENCES
Estoque(IDestoque)
);
-- dados
INSERT INTO Estoque (Nome, quantidade)
VALUES
('Coca-Cola 2L', 100),
('Arroz Branco 5kg', 50),
('Detergente Líquido 500ml', 200),
```

```
('Sabão em Pó 1kg', 75);

INSERT INTO Vendas (produto_estoqueID, quantidade)
VALUES
(1, 2),
(2, 1),
(1, 3),
(3, 5),
(4, 4);

SELECT * FROM Vendas;
--=========================================================
-- 6. Banco de Dados para Aplicação de Desconto(Exercício6)
-- =========================================================
CREATE DATABASE Lojadesconto;
USE Lojadesconto;

CREATE TABLE Produtos(
     IDprodutos INT AUTO_INCREMENT PRIMARY KEY,
     nome VARCHAR(100),
     preco DECIMAL(10,2),
     categoria VARCHAR(50)

);

INSERT INTO Produtos(nome, preco, categoria)
```

```
VALUES
('Jogo de Cama', 69.99, 'Casa'),
('Cafeteira Elétrica', 149.90, 'Eletrodomésticos'),
('Mochila Escolar', 89.99, 'Acessórios'),
('Fone de Ouvido Bluetooth', 199.50, 'Eletrônicos'),
('Tênis Esportivo', 299.99, 'Calçados'),
('Cadeira Gamer', 499.99, 'Móveis'),
('Smartphone Modelo X', 1999.00, 'Eletrônicos'),
('Arroz Parboilizado 5kg', 15.99, 'Alimentos'),
('Detergente Líquido 500ml', 3.49, 'Limpeza'),
('Geladeira Inox', 2599.90, 'Eletrodomésticos');

-- ====================================================
-- 7. Banco de Dados para Bloqueio de Alterações em Contratos (Exercício7)
-- ====================================================
CREATE DATABASE EmpresaContratos;
USE EmpresaContratos;

CREATE TABLE Contratos(
     IDcontratos INT AUTO_INCREMENT PRIMARY KEY,
     descricao TEXT,
     valor DECIMAL(10,2),
     Status ENUM('Aberto','Fechado')
);
```

```
INSERT INTO Contratos (descricao, valor, Status)
VALUES
('Contrato de fornecimento de materiais', 15000.00,
'Aberto'),
('Contrato de prestação de serviços de limpeza', 5000.50,
'Aberto'),
('Contrato de consultoria em TI', 25000.75, 'Fechado'),
('Contrato de locação de espaço', 12000.00, 'Aberto'),
('Contrato de manutenção predial', 8000.00, 'Fechado');

-- ======================================================
-- 8. Banco de Dados para Geração de Relatórios de Vendas (Exercício 8)
-- ======================================================
CREATE DATABASE RelatorioVendas;
USE RelatorioVendas;

CREATE TABLE Vendas(
     IDvendas INT AUTO_INCREMENT PRIMARY KEY,
     Nome VARCHAR(100),
     quantidade INT,
     valor_total DECIMAL(10,2),
     data_venda DATE
);

INSERT INTO Vendas (Nome, quantidade, valor_total,
data_venda)
```

```
VALUES
('Dell Inspiron', 10, 15000.00, '2024-01-15'),
('Samsung Galaxy', 5, 3750.00, '2024-02-20'),
('Fone de ouvido Sony ', 8, 4800.00, '2024-03-10'),
('Apple iPad', 12, 24000.00, '2024-04-05'),
('Câmera Canon', 20, 60000.00, '2024-05-15');

-- ======================================================
-- 9. Banco de Dados para Auditoria de Alterações de Usuários (Exercício 9)
-- ======================================================
CREATE DATABASE AuditoriaUsuarios;
USE AuditoriaUsuarios;

CREATE TABLE Usuarios(
     IDusuarios INT AUTO_INCREMENT PRIMARY KEY,
     nome VARCHAR(100),
     email VARCHAR(100),
     senha VARCHAR(100)
);

CREATE TABLE Usuarios_Auditoria(
     IDuser INT AUTO_INCREMENT PRIMARY KEY,
     usuarioID INT,
     tipo_alteracao ENUM('INSERT', 'UPDATE','DELETE'),
     data_alteracao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    FOREIGN KEY (usuarioID) REFERENCES
Usuarios(IDusuarios)


);


INSERT INTO Usuarios (nome, email, senha)
VALUES
('Ana Silva', 'ana.silva@email.com', 'senha123'),
('Carlos Souza', 'carlos.souza@email.com', 'senha456'),
('Mariana Oliveira', 'mariana.oliveira@email.com',
'senha789'),
('Lucas Santos', 'lucas.santos@email.com', 'senha101'),
('Fernanda Costa', 'fernanda.costa@email.com', 'senha202');


INSERT INTO Usuarios_Auditoria (usuarioID, tipo_alteracao)
VALUES
(1, 'INSERT'),
(2, 'INSERT'),
(1, 'UPDATE'),
(3, 'INSERT'),
(4, 'DELETE'),
(5, 'UPDATE');
============================================================
CREATE DATABASE Lojacreditos;
USE Lojacreditos;
```

```
CREATE TABLE Clientes (
     IDclientes INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    saldo_creditos DECIMAL(10,2)
);

CREATE TABLE Vendas(
     IDvendas INT AUTO_INCREMENT PRIMARY KEY,
    clienteID INT,
    valor_venda DECIMAL(10,2),
    data_venda TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (clienteID) REFERENCES Clientes(IDclientes)
);

INSERT INTO Clientes (nome, saldo_creditos) VALUES
('João Silva', 1500.00),
('Maria Oliveira', 2500.50),
('Carlos Santos', 500.75),
('Ana Costa', 1200.00);

INSERT INTO Vendas (clienteID, valor_venda) VALUES
(1, 300.00),
(2, 450.75),
(1, 150.50),
(3, 800.00),
```

```
(4, 250.00);
```

Exercícios

1. Criar uma Trigger para Atualizar um Registro

- **Descrição:** Crie uma trigger que seja executada automaticamente após a atualização de uma tabela de funcionários, de forma que sempre que o salário de um funcionário for alterado, a trigger atualize uma tabela de histórico de salários.

```
USE Empresa;


DELIMITER $$

CREATE TRIGGER up_funcionario

AFTER UPDATE ON funcionarios

FOR EACH ROW

BEGIN

     IF old.salario <> new.salario THEN

          INSERT INTO historico_salarios(funcionarioID,
salario_antigo, salario_novo)

          VALUES (new.IDfuncionario, old.salario, new.salario);

     END IF;

END $$

DELIMITER ;


SELECT * FROM Funcionarios;


UPDATE Funcionarios

SET salario = 8500

WHERE IDfuncionario = 3;
```

```
SELECT * FROM historico_salarios;
```

2. Criar uma Stored Procedure para Inserir Registros

- **Descrição:** Escreva uma stored procedure que insira um novo produto em uma tabela Produtos, recebendo parâmetros como nome, preço, e categoria. A procedure deve verificar se o nome já existe antes de inserir.

```
USE Lojadesconto;


-- CRIAÇÃO DA PROCEDURE

DELIMITER $$

CREATE PROCEDURE novo_produto(IN n_nome VARCHAR(100), IN n_preco
DECIMAL(10,2), IN n_categoria VARCHAR(50))

BEGIN

     IF EXISTS (SELECT 1 FROM Produtos WHERE nome = n_nome)
THEN

          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'O produto
já existe';

     ELSE

          INSERT INTO Produtos (nome, preco, categoria)

        VALUES(n_nome, n_preco, n_categoria);

     END IF;

END $$

DELIMITER ;


-- Chamar a procedure

CALL novo_produto('Jogo de Cama', 69.99, 'Casa');
```

```
-- 0  12    17:07:53   CALL novo_produto('Jogo de Cama', 69.99,
'Casa') Error Code: 1644. O produto já existe    0.578 sec
```

3. Criar uma Trigger para Registro de Exclusões

- **Descrição:** Crie uma trigger que, ao deletar um registro da tabela Clientes, mova as informações excluídas para uma tabela de backup chamada Clientes_Excluidos, juntamente com a data da exclusão.

```
USE Vendas_t;
 DELIMITER $$
 CREATE TRIGGER delete_object
 AFTER DELETE ON Clientes
 FOR EACH ROW
 BEGIN
     INSERT INTO Clientes_excluidos(nome, email, telefone)
    VALUES (old.nome, old.email, old.telefone);

 END $$
 DELIMITER ;

SELECT * FROM clientes;
SELECT * FROM clientes_excluidos;

DELETE FROM Clientes WHERE IDcliente = 2;
DELETE FROM Clientes WHERE IDcliente = 4;
```

4. Stored Procedure para Cálculo de Média

- **Descrição:** Crie uma stored procedure que receba como parâmetro o ID de um aluno e retorne a média das suas notas, considerando uma tabela Notas. A média deve ser calculada dinamicamente na procedure.

```
USE Escola_t;


DELIMITER $$

CREATE PROCEDURE calculaMedia (IN alunosID INT)

BEGIN

     DECLARE media DECIMAL(5,2);

     SELECT AVG(nota) INTO media FROM Notas WHERE alunosID =
alunosID;

     SELECT CONCAT('A media das notas do aluno cujo ID é =  ',
alunosID ,',  é  ', media) AS Resultado;

END$$

DELIMITER ;


-- DROP PROCEDURE IF EXISTS calculaMedia;

CALL calculaMedia(1);
```

5. Criar uma Trigger para Atualizar Estoque

- **Descrição:** Crie uma trigger que, após uma venda ser registrada na tabela Vendas, atualize a quantidade de produtos disponíveis no estoque na tabela Estoque.

```
SELECT * FROM Vendas;

SELECT * FROM Estoque;


DELIMITER $$

 CREATE TRIGGER atualizarEstoque

 AFTER INSERT ON estoque

 FOR EACH ROW

 BEGIN

    UPDATE estoque

    SET quantidade = quantidade - new.quantidade

    WHERE IDestoque = new.IDestoque;

END$$


DELIMITER ;
```

6. Stored Procedure para Aplicar Desconto

- **Descrição:** Crie uma stored procedure que aplique um desconto em todos os produtos de uma categoria específica. A procedure deve receber o nome da categoria e o percentual de desconto como parâmetros.

```
DELIMITER $$

CREATE PROCEDURE aplicar_desconto_categoria(

    IN p_categoria_nome VARCHAR(100),

    IN p_percentual_desconto DECIMAL(5,2) )

BEGIN

    UPDATE Produtos

    INNER JOIN Categorias

    ON Produtos.categoria_id = Categorias.id

    SET Produtos.preco = Produtos.preco - (Produtos.preco *
(p_percentual_desconto / 100))

    WHERE Categorias.nome = p_categoria_nome;


    -- Mensagem de confirmação da operação.
```

```
    SELECT CONCAT('Desconto de ', p_percentual_desconto, '%
aplicado a todos os produtos da categoria: ', p_categoria_nome)
AS Mensagem;


END$$

DELIMITER ;


CALL aplicar_desconto_categoria('Eletrônicos', 15.00);
```

7. Criar uma Trigger para Bloquear Alterações

- **Descrição:** Crie uma trigger que impeça qualquer alteração em uma tabela Contratos se o campo Status estiver marcado como "Fechado". A trigger deve cancelar a operação e exibir uma mensagem de erro.

```
USE Empresacontratos;


SELECT * FROM contratos;


DELIMITER $$

CREATE TRIGGER bloqueio

BEFORE UPDATE ON contratos

FOR EACH ROW

BEGIN

     IF old.status = 'Fechado' THEN

          SIGNAL SQLSTATE '45000'

          SET MESSAGE_TEXT = 'Não é possível alterar o
contrato';
```

```
     END IF;
END $$
DELIMITER ;


UPDATE contratos
SET valor = 100
WHERE IDcontratos = 3;


-- Resposta
-- 0  7    08:45:52   UPDATE contratos  SET valor = 100  WHERE
IDcontratos = 3  Error Code: 1644. Não é possível alterar o
contrato   0.000 sec
```

8. Stored Procedure para Geração de Relatórios

- **Descrição:** Escreva uma stored procedure que gere um relatório com a quantidade de vendas realizadas em cada mês. A procedure deve retornar o mês, a quantidade de vendas e o valor total vendido.

```
DELIMITER $$
CREATE PROCEDURE relatorio_venda()
     SELECT DATE_FORMAT(data_venda, '%M') as mes,
SUM(quantidade), SUM(valor_total)
     FROM vendas where data_venda
     GROUP BY mes;


END $$
DELIMITER ;
CALL relatorio_venda();
```

9. Criar uma Trigger para Auditoria

- **Descrição:** Crie uma trigger que grave todas as alterações realizadas na tabela Usuarios em uma tabela de auditoria chamada Usuarios_Auditoria. A trigger deve registrar o usuário que fez a alteração, o tipo de alteração (INSERT, UPDATE ou DELETE), e a data/hora.

```
DELIMITER $$


CREATE TRIGGER auditoria_users

AFTER INSERT ON usuarios

FOR EACH ROW

BEGIN

    INSERT INTO usuarios_auditoria (usuariosID,
tipo_alteracao)

    VALUES (new.IDusuarios, 'INSERT');

END $$
```

```
CREATE TRIGGER auditoria_users_up

AFTER UPDATE ON usuarios

FOR EACH ROW

BEGIN

    INSERT INTO usuarios_auditoria (usuariosID,
tipo_alteracao)

    VALUES (new.IDusuarios, 'UPDATE');

END $$


CREATE TRIGGER auditoria_users_del

AFTER DELETE ON usuarios

FOR EACH ROW

BEGIN

    INSERT INTO usuarios_auditoria (usuariosID,
tipo_alteracao)

    VALUES (old.IDusuarios, 'DELETE');

END $$


DELIMITER ;
```

10. Stored Procedure para Verificação de Créditos

- **Descrição:** Escreva uma stored procedure que verifique o saldo de créditos de um cliente em uma tabela Clientes antes de permitir que uma venda seja registrada. Se o saldo for inferior ao valor da compra, a procedure deve retornar uma mensagem de erro.

```
DELIMITER $$
```

```
CREATE PROCEDURE registro_venda (IN clienteID INT, IN valorVenda
DECIMAL(10,2))

BEGIN

    DECLARE saldoAtual DECIMAL(10,2);

    SELECT saldo_creditos INTO saldoAtual

    FROM Clientes WHERE IDclientes = clienteID;


    IF saldoAtual IS NULL THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cliente não
encontrado.';

    END IF;


    IF saldoAtual < valorVenda THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Saldo
insuficiente ';

    ELSE

        INSERT INTO Vendas (clienteID, valor_venda) VALUES
(clienteID, valorVenda);


        UPDATE Clientes

        SET saldo_creditos = saldo_creditos - valorVenda

        WHERE IDclientes = clienteID;

    END IF;

END $$


DELIMITER ;
```