```
#Tabelas:
CREATE DATABASE banco_neb;
USE banco_neb;

-- Tabela de Cliente Pessoa Física
CREATE TABLE cliente_pf (
    id_cliente_pf INT AUTO_INCREMENT PRIMARY KEY,
    nome_cliente_pf VARCHAR(150) NOT NULL,
    data_de_nascimento DATE NOT NULL,
    cpf VARCHAR(11) UNIQUE NOT NULL,
    telefone VARCHAR(11) NOT NULL,
    email VARCHAR(150) NOT NULL,
    senha_de_entrada VARCHAR(8) NOT NULL,
    senha_de_autorizacao VARCHAR(8) NOT NULL,
    saldo DECIMAL(10,2) NOT NULL DEFAULT 0
);

-- Tabela de Endereço Cliente Pessoa Física
CREATE TABLE endereco_pf (
    id_endereco_pf INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente INT NOT NULL,
    logradouro VARCHAR(150) NOT NULL,
    bairro VARCHAR(150) NOT NULL,
    cidade VARCHAR(150) NOT NULL,
    estado VARCHAR(150) NOT NULL,
    pais VARCHAR(150) NOT NULL,
    referencia VARCHAR(150) NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES cliente_pf(id_cliente_pf)
);

-- Tabela de Extrato Cliente Pessoa Física
CREATE TABLE extrato_pf (
    id_extrato_pf INT AUTO_INCREMENT PRIMARY KEY,
    id_pagador INT NOT NULL,
    valor_transacao DECIMAL(10,2) NOT NULL,
    tipo_pagamento ENUM('Crédito', 'Débito'),
    num_parcela_atual INT,
    num_parcela_total INT,
    id_beneficiario INT,
    data_transacao DATETIME NOT NULL,
    FOREIGN KEY (id_pagador) REFERENCES cliente_pf(id_cliente_pf),
```

```
    FOREIGN KEY (id_beneficiario) REFERENCES
cliente_pf(id_cliente_pf)
);

-- Tabela de Crédito Cliente Pessoa Física
CREATE TABLE credito_pf (
    id_credito_pf INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente_credito_pf INT NOT NULL,
    valor_disponivel DECIMAL(10,2) NOT NULL DEFAULT 50.00,
    valor_total DECIMAL(10,2) NOT NULL DEFAULT 50.00,
    FOREIGN KEY (id_cliente_credito_pf) REFERENCES
cliente_pf(id_cliente_pf)
);

-- Tabela de Empréstimo Cliente Pessoa Física
CREATE TABLE emprestimo_pf (
    id_emprestimo_pf INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente_emprestimo_pf INT NOT NULL,
    valor_pendente DECIMAL(10,2) NOT NULL DEFAULT 0.00,
    valor_total_disponivel DECIMAL(10,2) NOT NULL DEFAULT 1000.00,
    status_emprestimo ENUM('Disponível', 'Pendente') DEFAULT
'Disponível',
    FOREIGN KEY (id_cliente_emprestimo_pf) REFERENCES
cliente_pf(id_cliente_pf)
);


-- Tabela de Lista de Empréstimos Cliente Pessoa Física
CREATE TABLE lista_emprestimo_pf (
    id_lista_emprestimo INT AUTO_INCREMENT PRIMARY KEY,
    id_emprestimo_pf INT NOT NULL,
    valor_total_emprestimo DECIMAL(10,2) NOT NULL,
    num_parcela INT NOT NULL,
    data_emprestimo DATE NOT NULL,
    FOREIGN KEY (id_emprestimo_pf ) REFERENCES emprestimo_pf
(id_emprestimo_pf )
);

-- Tabela de Pagamento Pendente Cliente Pessoa Física
CREATE TABLE pagamento_pendente_pf (
    id_pagamento_pendente_pf INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente_pf_pagamento INT NOT NULL,
```

```
    id_lista_emprestimo INT,
    valor_total_mes DECIMAL(10,2) NOT NULL,
    mes_parcela INT NOT NULL,
    ano_parcela INT NOT NULL,
    tipo_de_pagamento ENUM('Crédito', 'Empréstimo') NOT NULL,
    status_pagamento ENUM('Pago', 'Pendente') DEFAULT 'Pendente',
    FOREIGN KEY (id_lista_emprestimo) REFERENCES
lista_emprestimo_pf(id_lista_emprestimo),
    FOREIGN KEY (id_cliente_pf_pagamento) REFERENCES
cliente_pf(id_cliente_pf)
);

-- Tabela de Lista de Parcelas Cliente Pessoa Física
CREATE TABLE lista_parcelas_pf (
    id_lista_parcela INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente_pf_lista_parcela INT NOT NULL,
    id_extrato_lista INT NOT NULL,
    valor_parcela_mes DECIMAL(10,2) NOT NULL,
    mes_parcela INT NOT NULL,
    ano_parcela INT NOT NULL,
    tipo_de_pagamento ENUM('Crédito', 'Empréstimo') NOT NULL,
    parcela_atual INT NOT NULL,
    FOREIGN KEY (id_cliente_pf_lista_parcela) REFERENCES
cliente_pf(id_cliente_pf),
 FOREIGN KEY (id_extrato_lista) REFERENCES extrato_pf(id_extrato_pf)


);


-- Tabela de Funcionário
CREATE TABLE funcionario (
    id_funcionario INT AUTO_INCREMENT PRIMARY KEY,
    nome_funcionario VARCHAR(150) NOT NULL,
    data_de_nascimento DATE NOT NULL,
    cpf VARCHAR(11) UNIQUE NOT NULL,
    telefone VARCHAR(11) NOT NULL,
    email VARCHAR(150) NOT NULL,
    usuario VARCHAR(8) NOT NULL UNIQUE,
    senha VARCHAR(8) NOT NULL,
    data_contratacao DATE NOT NULL,
```

```
    cargo VARCHAR(150) NOT NULL,
    salario DECIMAL(10,2),
    status_funcionario ENUM('Ativo', 'Licença', 'Desligado') DEFAULT
'Ativo'
);

-- Tabela de Endereço do Funcionário
CREATE TABLE endereco_funcionario (
    id_endereco_funcionario INT AUTO_INCREMENT PRIMARY KEY,
    id_funcionario INT NOT NULL,
    logradouro VARCHAR(150) NOT NULL,
    bairro VARCHAR(150) NOT NULL,
    cidade VARCHAR(150) NOT NULL,
    estado VARCHAR(150) NOT NULL,
    pais VARCHAR(150) NOT NULL,
    referencia VARCHAR(150) NOT NULL,
    FOREIGN KEY (id_funcionario) REFERENCES
funcionario(id_funcionario)
);

-- Tabela de Registro de Modificação de Dados do Cliente
CREATE TABLE registro_modificacao_dados_cliente_pf (
    id_registro INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente_registro INT,
    coluna_modificada
ENUM('Nome','Telefone','Email','senha_de_entrada',
'senha_de_autorizacao'),
    registro_antigo VARCHAR(150),
    registro_novo VARCHAR(150),
    data_modificacao DATE NOT NULL,
    FOREIGN KEY (id_cliente_registro) REFERENCES
cliente_pf(id_cliente_pf)
);

-- Tabela de Modificação de Endereço do Cliente
CREATE TABLE modificar_endereco_cliente_pf (
    id_registro_antigo INT AUTO_INCREMENT PRIMARY KEY,
    id_endereco_pf INT,
    logradouro_antigo VARCHAR(150) NOT NULL,
    bairro_antigo VARCHAR(150) NOT NULL,
    cidade_antigo VARCHAR(150) NOT NULL,
```

```
    estado_antigo VARCHAR(150) NOT NULL,
    pais_antigo VARCHAR(150) NOT NULL,
    referencia_antigo VARCHAR(150) NOT NULL,
    data_modificacao DATE NOT NULL,
    FOREIGN KEY (id_endereco_pf  ) REFERENCES endereco_pf
(id_endereco_pf )
);


-- Tabela de Registro de Modificação de Dados do Funcionário
CREATE TABLE registro_modificacao_dados_funcionario (
    id_registro_funcionario INT AUTO_INCREMENT PRIMARY KEY,
    id_funcionario_registro INT,
    coluna_modificada
ENUM('Nome','Telefone','Email','Senha','Cargo','Salario','Status_fun
cionario'),
    registro_antigo VARCHAR(150),
    registro_novo VARCHAR(150),
    data_modificacao DATE NOT NULL,
    FOREIGN KEY (id_funcionario_registro) REFERENCES
funcionario(id_funcionario)
);

-- Tabela de Modificação de Endereço do Funcionário
CREATE TABLE modificar_endereco_funcionario (
    id_registro_antigo INT AUTO_INCREMENT PRIMARY KEY,
    id_endereco_funcionario INT,
    logradouro_antigo VARCHAR(150) NOT NULL,
    bairro_antigo VARCHAR(150) NOT NULL,
    cidade_antigo VARCHAR(150) NOT NULL,
    estado_antigo VARCHAR(150) NOT NULL,
    pais_antigo VARCHAR(150) NOT NULL,
    referencia_antigo VARCHAR(150) NOT NULL,
    data_modificacao DATE NOT NULL,
    FOREIGN KEY (id_endereco_funcionario ) REFERENCES
endereco_funcionario (id_endereco_funcionario )
);

-- Tabela de Gestão de Juros
CREATE TABLE gestao_juros (
    juros_credito DECIMAL(10,2) NOT NULL,
```

```
    juros_emprestimo DECIMAL(10,2) NOT NULL
);
CREATE TABLE negoc_novos_cred_empre(
     id_negoc INT AUTO_INCREMENT PRIMARY KEY,
     id_gerente INT NOT NULL,
     id_cliente_negoc INT NOT NULL,
     tipo_financiamento ENUM('Crédito','Empréstimo') NOT NULL,
     valor_financiamento_antigo DECIMAL(10.2) NOT NULL,
     valor_financiamento_novo DECIMAL(10.2) NOT NULL,
     data_modif DATE NOT NULL,
     FOREIGN KEY (id_gerente) REFERENCES
     funcionario(id_funcionario),
     FOREIGN KEY (id_cliente_negoc) REFERENCES
     cliente_pf(id_cliente_pf));


-- Inserção de Juros Padrão
INSERT INTO gestao_juros (juros_credito, juros_emprestimo) VALUES
(2.00, 5.00);
```

```
#Procedures:

DELIMITER $$

CREATE PROCEDURE inserir_cliente_e_dados(
    input_nome_cliente_pf VARCHAR(150),
    input_data_de_nascimento DATE,
    input_cpf VARCHAR(11),
    input_telefone VARCHAR(11),
    input_email VARCHAR(150),
    input_senha_de_entrada VARCHAR(8),
    input_senha_de_autorizacao VARCHAR(8),
    input_logradouro VARCHAR(150),
    input_bairro VARCHAR(150),
    input_cidade VARCHAR(150),
    input_estado VARCHAR(150),
    input_pais VARCHAR(150),
    input_referencia VARCHAR(150)
)
BEGIN
    DECLARE novo_id_cliente_pf INT;

    INSERT INTO cliente_pf(nome_cliente_pf, data_de_nascimento, cpf,
telefone, email, senha_de_entrada, senha_de_autorizacao)
    VALUES(input_nome_cliente_pf, input_data_de_nascimento,
input_cpf, input_telefone, input_email, input_senha_de_entrada,
input_senha_de_autorizacao);

    SET novo_id_cliente_pf = LAST_INSERT_ID();

    INSERT INTO endereco_pf(id_cliente, logradouro, bairro, cidade,
estado, pais, referencia)
    VALUES(novo_id_cliente_pf, input_logradouro, input_bairro,
input_cidade, input_estado, input_pais, input_referencia);

    INSERT INTO credito_pf(id_cliente_credito_pf)
VALUES(novo_id_cliente_pf);

    INSERT INTO emprestimo_pf(id_cliente_emprestimo_pf)
VALUES(novo_id_cliente_pf);
```

```
END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE registro_transacao(
    input_id_cliente_pf INT,
    input_id_beneficiario INT,
    input_valor_transacao DECIMAL(10, 2),
    input_tipo_transacao ENUM('Crédito', 'Débito'),
    input_num_parcelas INT
)
BEGIN
    DECLARE ano_transacao INT;
    DECLARE mes_transacao INT;
    DECLARE id_lista_pagamento INT;
    DECLARE contador INT;
    DECLARE id_extrato INT;
    DECLARE valor_parcela DECIMAL(10, 2);
    DECLARE quant_parcela INT;

    SET ano_transacao = YEAR(CURDATE());
    SET mes_transacao = MONTH(CURDATE());
    SET quant_parcela = input_num_parcelas;
    SET contador = 1;

    UPDATE cliente_pf
    SET saldo = saldo + input_valor_transacao
    WHERE id_cliente_pf = input_id_beneficiario;

    IF input_tipo_transacao = 'Débito' THEN
        UPDATE cliente_pf
        SET saldo = saldo - input_valor_transacao
        WHERE id_cliente_pf = input_id_cliente_pf;

        INSERT INTO extrato_pf(id_pagador, valor_transacao,
tipo_pagamento, id_beneficiario, data_transacao)
        VALUES(input_id_cliente_pf, input_valor_transacao,
input_tipo_transacao, input_id_beneficiario, NOW());
```

```
    ELSEIF input_tipo_transacao = 'Crédito' THEN
        SET valor_parcela = calcular_parcelas(input_valor_transacao,
input_num_parcelas, input_tipo_transacao);

        UPDATE credito_pf
        SET valor_disponivel = valor_disponivel - (valor_parcela *
input_num_parcelas)
        WHERE id_cliente_credito_pf = input_id_cliente_pf;

        INSERT INTO extrato_pf(id_pagador, valor_transacao,
tipo_pagamento, num_parcela_total, id_beneficiario, data_transacao)
        VALUES(input_id_cliente_pf, input_valor_transacao,
input_tipo_transacao, input_num_parcelas, input_id_beneficiario,
NOW());

        SET id_extrato = LAST_INSERT_ID();

        WHILE quant_parcela > 0 DO
            SET id_lista_pagamento =
verificar_registro_credito(ano_transacao, mes_transacao,
input_id_cliente_pf, input_tipo_transacao);

            INSERT INTO
lista_parcelas_pf(id_cliente_pf_lista_parcela, id_extrato_lista ,
valor_parcela_mes, mes_parcela, ano_parcela, tipo_de_pagamento,
parcela_atual)
            VALUES(input_id_cliente_pf, id_extrato, valor_parcela,
mes_transacao, ano_transacao, input_tipo_transacao, contador);

            IF id_lista_pagamento IS NULL THEN
                INSERT INTO pagamento_pendente_pf
(id_cliente_pf_pagamento, valor_total_mes, mes_parcela, ano_parcela,
tipo_de_pagamento)
                VALUES(input_id_cliente_pf, valor_parcela,
mes_transacao, ano_transacao, input_tipo_transacao);
            ELSE
                UPDATE pagamento_pendente_pf
                SET valor_total_mes = valor_total_mes +
valor_parcela
                WHERE id_pagamento_pendente_pf = id_lista_pagamento;
            END IF;
```

```
            IF mes_transacao = 12 THEN
                SET mes_transacao = 1;
                SET ano_transacao = ano_transacao + 1;
            ELSE
                SET mes_transacao = mes_transacao + 1;
            END IF;

            SET quant_parcela = quant_parcela - 1;
            SET contador = contador + 1;
        END WHILE;
    END IF;
END$$

DELIMITER ;



DELIMITER $$

CREATE PROCEDURE negociacao_emprestimo(
    input_id_emprestimo INT,
    input_id_cliente_pf INT,
    input_valor_total_emprestimo DECIMAL(10,2),
    input_quant_parcelas INT
)
BEGIN
    DECLARE valor_parcela DECIMAL(10,2);
    DECLARE quant_parcela INT;
    DECLARE ano_transacao INT;
    DECLARE mes_transacao INT;
    DECLARE valor_total_emprestimo DECIMAL(10,2);
    DECLARE id_emprestimo INT;

    SET ano_transacao = YEAR(CURDATE());
    SET mes_transacao = MONTH(CURDATE());
    SET quant_parcela = input_quant_parcelas;
    SET valor_parcela =
calcular_parcelas(input_valor_total_emprestimo,
input_quant_parcelas, 'Empréstimo');
```

```
    SET valor_total_emprestimo = valor_parcela *
input_quant_parcelas;

    -- Atualização de saldo do cliente
    UPDATE cliente_pf
    SET saldo = saldo + input_valor_total_emprestimo
    WHERE id_cliente_pf = input_id_cliente_pf;

    -- Atualização do empréstimo
    UPDATE emprestimo_pf
    SET valor_pendente = valor_total_emprestimo,
        status_emprestimo = 'Pendente'
    WHERE id_emprestimo_pf = input_id_emprestimo;

    -- Inserção na lista de empréstimos
    INSERT INTO lista_emprestimo_pf(id_emprestimo_pf,
valor_total_emprestimo, num_parcela, data_emprestimo)
    VALUES(input_id_emprestimo, valor_total_emprestimo,
input_quant_parcelas, NOW());

    SET id_emprestimo = LAST_INSERT_ID();

    -- Loop para gerar parcelas
    WHILE quant_parcela > 0 DO
        INSERT INTO pagamento_pendente_pf (id_cliente_pf_pagamento,
id_lista_emprestimo, valor_total_mes, mes_parcela, ano_parcela,
tipo_de_pagamento)
        VALUES(input_id_cliente_pf, id_emprestimo, valor_parcela,
mes_transacao, ano_transacao, 'Empréstimo');

        -- Atualização do mês e ano
        IF mes_transacao = 12 THEN
            SET mes_transacao = 1;
            SET ano_transacao = ano_transacao + 1;
        ELSE
            SET mes_transacao = mes_transacao + 1;
        END IF;

        -- Reduz o contador de parcelas
        SET quant_parcela = quant_parcela - 1;
    END WHILE;
```

```
END$$

DELIMITER ;



DELIMITER $$

CREATE PROCEDURE pagamento_parcela_pendente(
    input_id_cliente_pf INT,
    input_valor_pago DECIMAL(10,2),
    input_id_pagamento_pendente INT,
    input_tipo_pagamento ENUM('Crédito', 'Empréstimo')
)
BEGIN
    DECLARE ano_transacao INT;
    DECLARE mes_transacao INT;
    DECLARE veric_valor_pendente DECIMAL(10,2);
    DECLARE veric_valor_pendente_mensal DECIMAL(10,2);

    SET ano_transacao = YEAR(CURDATE());
    SET mes_transacao = MONTH(CURDATE());

    UPDATE cliente_pf
    SET saldo = saldo - input_valor_pago
    WHERE id_cliente_pf = input_id_cliente_pf;

    IF input_tipo_pagamento = 'Crédito' THEN
        UPDATE credito_pf
        SET valor_disponivel = valor_disponivel + input_valor_pago
        WHERE id_cliente_credito_pf = input_id_cliente_pf;
    ELSE
        UPDATE emprestimo_pf
        SET valor_pendente = valor_pendente - input_valor_pago
        WHERE id_cliente_emprestimo_pf = input_id_cliente_pf;

        SELECT valor_pendente INTO veric_valor_pendente
        FROM emprestimo_pf
        WHERE id_cliente_emprestimo_pf = input_id_cliente_pf;

        IF veric_valor_pendente = 0 THEN
            UPDATE emprestimo_pf
```

```
            SET status_emprestimo = 'Disponível'
            WHERE id_cliente_emprestimo_pf = input_id_cliente_pf;
        END IF;
    END IF;

    INSERT INTO extrato_pf(id_pagador, valor_transacao,
tipo_pagamento, data_transacao)
    VALUES(input_id_cliente_pf, input_valor_pago, 'Débito', NOW());

    UPDATE pagamento_pendente_pf
    SET valor_total_mes = valor_total_mes - input_valor_pago
    WHERE id_pagamento_pendente_pf = input_id_pagamento_pendente;

    SELECT valor_total_mes INTO veric_valor_pendente_mensal
    FROM pagamento_pendente_pf
    WHERE id_pagamento_pendente_pf = input_id_pagamento_pendente;

    IF veric_valor_pendente_mensal = 0 THEN
        UPDATE pagamento_pendente_pf
        SET status_pagamento = 'PAGO'
        WHERE id_pagamento_pendente_pf =
input_id_pagamento_pendente;
    END IF;
END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE inserir_funcionario(
    input_nome_funcionario VARCHAR(150),
    input_data_de_nascimento DATE,
    input_cpf VARCHAR(11),
    input_telefone VARCHAR(11),
    input_email VARCHAR(150),
    input_usuario VARCHAR(8),
    input_senha VARCHAR(8),
    input_cargo VARCHAR(150),
    input_salario DECIMAL(10,2),
    input_logradouro VARCHAR(150),
    input_bairro VARCHAR(150),
```

```
    input_cidade VARCHAR(150),
    input_estado VARCHAR(150),
    input_pais VARCHAR(150),
    input_referencia VARCHAR(150)
)
BEGIN
    DECLARE novo_id_funcionario INT;

    INSERT INTO funcionario(
        nome_funcionario,
        data_de_nascimento,
        cpf,
        telefone,
        email,
        usuario,
        senha,
        data_contratacao,
        cargo,
        salario
    )
    VALUES(
        input_nome_funcionario,
        input_data_de_nascimento,
        input_cpf,
        input_telefone,
        input_email,
        input_usuario,
        input_senha,
        NOW(),
        input_cargo,
        input_salario
    );

    SET novo_id_funcionario = LAST_INSERT_ID();

    INSERT INTO endereco_funcionario(
        id_funcionario,
        logradouro,
        bairro,
        cidade,
        estado,
```

```
        pais,
        referencia
    )
    VALUES(
        novo_id_funcionario,
        input_logradouro,
        input_bairro,
        input_cidade,
        input_estado,
        input_pais,
        input_referencia
    );

END $$

DELIMITER ;



DELIMITER $$

CREATE PROCEDURE modific_finan_cliente_pf(
    input_id_gerente INT,
    input_id_cliente_negoc INT,
    input_tipo_financiamento ENUM('Crédito', 'Empréstimo'),
    input_valor_financiamento_novo DECIMAL(10,2)
)
BEGIN
    DECLARE input_valor_financiamento_antigo DECIMAL(10,2);

    -- Inicializando a variável com um valor padrão
    SET input_valor_financiamento_antigo = 0.00;

    IF input_tipo_financiamento = 'Crédito' THEN
        -- Atribuindo o valor antigo à variável de forma correta
        SELECT valor_total INTO input_valor_financiamento_antigo
        FROM credito_pf
        WHERE id_cliente_credito_pf = input_id_cliente_negoc;

        -- Caso o SELECT não retorne nada (ou seja, não encontre o
cliente), usa o valor default
```

```
        IF input_valor_financiamento_antigo IS NULL THEN
            SET input_valor_financiamento_antigo = 0.00; -- valor
padrão
        END IF;

        -- Atualizando o valor do financiamento
        UPDATE credito_pf
        SET valor_total = input_valor_financiamento_novo
        WHERE id_cliente_credito_pf = input_id_cliente_negoc;

        -- Inserindo a alteração no histórico
        INSERT INTO negoc_novos_cred_empre (
            id_gerente, id_cliente_negoc, tipo_financiamento,
            valor_financiamento_antigo, valor_financiamento_novo,
data_modif
        )
        VALUES (input_id_gerente, input_id_cliente_negoc, 'Crédito',
input_valor_financiamento_antigo, input_valor_financiamento_novo,
CURDATE());

    ELSEIF input_tipo_financiamento = 'Empréstimo' THEN
        -- Atribuindo o valor antigo à variável de forma correta
        SELECT valor_total_disponivel INTO
input_valor_financiamento_antigo
        FROM emprestimo_pf
        WHERE id_cliente_emprestimo_pf = input_id_cliente_negoc;

        -- Caso o SELECT não retorne nada (ou seja, não encontre o
cliente), usa o valor default
        IF input_valor_financiamento_antigo IS NULL THEN
            SET input_valor_financiamento_antigo = 0.00; -- valor
padrão
        END IF;

        -- Atualizando o valor do empréstimo
        UPDATE emprestimo_pf
        SET valor_total_disponivel = input_valor_financiamento_novo
        WHERE id_cliente_emprestimo_pf = input_id_cliente_negoc;

        -- Inserindo a alteração no histórico
        INSERT INTO negoc_novos_cred_empre (
```

```
            id_gerente, id_cliente_negoc, tipo_financiamento,
            valor_financiamento_antigo, valor_financiamento_novo,
data_modif
        )
        VALUES (input_id_gerente, input_id_cliente_negoc,
'Empréstimo', input_valor_financiamento_antigo,
input_valor_financiamento_novo, CURDATE());
    END IF;

END $$

DELIMITER ;









#Funções:

DELIMITER $$
```

```
CREATE FUNCTION verifica_saldo(id_cliente INT, valor_transferencia
DECIMAL(10,2))
RETURNS BOOLEAN
READS SQL DATA
BEGIN
    DECLARE saldo_atual DECIMAL(10,2);

    SELECT saldo INTO saldo_atual
    FROM cliente_pf
    WHERE id_cliente_pf = id_cliente;

    RETURN valor_transferencia <= saldo_atual;
END $$

DELIMITER ;


DELIMITER $$

CREATE FUNCTION verificar_credencias(cpf_veric VARCHAR(11),
senha_veric VARCHAR(8))
RETURNS BOOLEAN
READS SQL DATA
BEGIN
    DECLARE senha_atual VARCHAR(8);

    SELECT senha_de_autorizacao INTO senha_atual
    FROM cliente_pf
    WHERE cpf = cpf_veric;

    RETURN senha_atual = senha_veric;
END $$

DELIMITER ;


DELIMITER $$

CREATE FUNCTION calcular_parcelas(
    valor_compra DECIMAL(10,2),
    num_parcelas INT,
```

```
    tipo_pagamento ENUM('Crédito', 'Empréstimo')
)
RETURNS DECIMAL(10,2)
READS SQL DATA
BEGIN
    DECLARE juros_atual DECIMAL(10,2);
    DECLARE valor_parcela DECIMAL(10,2);

    IF tipo_pagamento = 'Crédito' THEN
        SELECT juros_credito INTO juros_atual
        FROM gestao_juros;
    ELSE
        SELECT juros_emprestimo INTO juros_atual
        FROM gestao_juros;
    END IF;

    SET juros_atual = juros_atual / 100;

    SET valor_parcela = valor_compra * (juros_atual * POWER(1 +
juros_atual, num_parcelas))
                        / (POWER(1 + juros_atual, num_parcelas) -
1);

    RETURN valor_parcela;
END $$

DELIMITER ;


DELIMITER $$

CREATE FUNCTION verificar_registro_credito(ano INT, mes INT,
id_cliente INT, tipo ENUM('Crédito', 'Empréstimo'))
RETURNS INT
READS SQL DATA
BEGIN
    DECLARE id_pagamento INT;

    SELECT id_pagamento_pendente_pf INTO id_pagamento
    FROM pagamento_pendente_pf
    WHERE id_cliente_pf_pagamento = id_cliente
```

```
      AND ano_parcela = ano
      AND mes_parcela = mes
      AND tipo_de_pagamento = tipo;

    RETURN id_pagamento;
END $$

DELIMITER ;
```

```
#Triggers:

DELIMITER $$
```

```
CREATE TRIGGER registro_modif_cliente_pf
BEFORE UPDATE
ON cliente_pf
FOR EACH ROW
BEGIN
    IF OLD.nome_cliente_pf  <> NEW.nome_cliente_pf THEN
        INSERT INTO
registro_modificacao_dados_cliente_pf(id_cliente_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_cliente_pf, "Nome", OLD.nome_cliente_pf,
NEW.nome_cliente_pf, NOW());
    END IF;

    IF OLD.telefone  <> NEW.telefone  THEN
        INSERT INTO
registro_modificacao_dados_cliente_pf(id_cliente_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_cliente_pf, "Telefone", OLD.telefone,
NEW.telefone, NOW());
    END IF;

    IF OLD.email  <> NEW.email  THEN
        INSERT INTO
registro_modificacao_dados_cliente_pf(id_cliente_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_cliente_pf, "Email", OLD.email, NEW.email,
NOW());
    END IF;

    IF OLD.senha_de_entrada  <> NEW.senha_de_entrada  THEN
        INSERT INTO
registro_modificacao_dados_cliente_pf(id_cliente_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_cliente_pf, "Senha de entrada",
OLD.senha_de_entrada, NEW.senha_de_entrada, NOW());
    END IF;

    IF OLD.senha_de_autorizacao  <> NEW.senha_de_autorizacao  THEN
        INSERT INTO
registro_modificacao_dados_cliente_pf(id_cliente_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
```

```
        VALUES(NEW.id_cliente_pf, "Senha de autorização",
OLD.senha_de_autorizacao, NEW.senha_de_autorizacao, NOW());
    END IF;
END$$
DELIMITER ;



DELIMITER $$

CREATE TRIGGER registro_modif_endereco_cliente_pf
BEFORE UPDATE
ON endereco_pf
FOR EACH ROW
BEGIN
    INSERT INTO modificar_endereco_cliente_pf(id_endereco_pf ,
logradouro_antigo, bairro_antigo, cidade_antigo, estado_antigo,
pais_antigo, referencia_antigo, data_modificacao)
    VALUES(NEW.id_endereco_pf, OLD.logradouro, OLD.bairro,
OLD.cidade, OLD.estado, OLD.pais, OLD.referencia, NOW());
END$$
DELIMITER ;

DELIMITER $$

CREATE TRIGGER registro_modif_funcionario
BEFORE UPDATE
ON funcionario
FOR EACH ROW
BEGIN
    IF OLD.nome_funcionario <> NEW.nome_funcionario  THEN
        INSERT INTO
registro_modificacao_dados_funcionario(id_funcionario_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_funcionario, "Nome", OLD.nome_funcionario,
NEW.nome_funcionario, NOW());
    END IF;

    IF OLD.telefone  <> NEW.telefone  THEN
```

```
        INSERT INTO
registro_modificacao_dados_funcionario(id_funcionario_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_funcionario, "telefone", OLD.telefone,
NEW.telefone, NOW());
    END IF;

    IF OLD.email  <> NEW.email  THEN
        INSERT INTO
registro_modificacao_dados_funcionario(id_funcionario_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_funcionario, "email", OLD.email, NEW.email,
NOW());
    END IF;

    IF OLD.senha <> NEW.senha THEN
        INSERT INTO
registro_modificacao_dados_funcionario(id_funcionario_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_funcionario, "senha", OLD.senha, NEW.senha,
NOW());
    END IF;

    IF OLD.cargo  <> NEW.cargo  THEN
        INSERT INTO
registro_modificacao_dados_funcionario(id_funcionario_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_funcionario, "cargo", OLD.cargo, NEW.cargo,
NOW());
    END IF;

    IF OLD.salario  <> NEW.salario  THEN
        INSERT INTO
registro_modificacao_dados_funcionario(id_funcionario_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_funcionario, "salario", OLD.salario,
NEW.salario, NOW());
    END IF;

    IF OLD.status_funcionario  <> NEW.status_funcionario   THEN
```

```
        INSERT INTO
registro_modificacao_dados_funcionario(id_funcionario_registro
,coluna_modificada ,registro_antigo ,registro_novo,data_modificacao)
        VALUES(NEW.id_funcionario, "status_funcionario",
OLD.status_funcionario, NEW.status_funcionario, NOW());
    END IF;
END$$
DELIMITER ;

DELIMITER $$

CREATE TRIGGER registro_modif_endereco_funcionario
BEFORE UPDATE
ON endereco_funcionario
FOR EACH ROW
BEGIN
    INSERT INTO
modificar_endereco_funcionario(id_endereco_funcionario ,
logradouro_antigo, bairro_antigo, cidade_antigo, estado_antigo,
pais_antigo, referencia_antigo, data_modificacao)
    VALUES(NEW.id_endereco_funcionario , OLD.logradouro, OLD.bairro,
OLD.cidade, OLD.estado, OLD.pais, OLD.referencia, NOW());
END$$
DELIMITER ;
```