#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(13) 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
);


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)
);


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 DATE NOT NULL,
    hora_transacao TIME NOT NULL,
    FOREIGN KEY (id_pagador) REFERENCES
cliente_pf(id_cliente_pf),
    FOREIGN KEY (id_beneficiario) REFERENCES
cliente_pf(id_cliente_pf)
);


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)
);


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)
);



CREATE TABLE lista_emprestimo_pf (
    id_lista_emprestimo INT AUTO_INCREMENT PRIMARY KEY,
    id_emprestimo_pf INT NOT NULL,
    num_parcela INT NOT NULL,
    data_emprestimo DATE NOT NULL,
    valor_pago DECIMAL(10,2) NOT NULL,
```

```
    valor_total_emprestimo DECIMAL(10,2) NOT NULL,
    status_emprestimo ENUM('Pago', 'Pendente') NOT NULL,
    FOREIGN KEY (id_emprestimo_pf ) REFERENCES emprestimo_pf
(id_emprestimo_pf )
);


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)
);


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)


);
```

```
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)
);



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 )
);


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(13) 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'
);



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)
);


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','Stat
us_funcionario'),
    registro_antigo VARCHAR(150),
    registro_novo VARCHAR(150),
    data_modificacao DATE NOT NULL,
    FOREIGN KEY (id_funcionario_registro) REFERENCES
funcionario(id_funcionario)
);
```

```
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 )
);
```

```
CREATE TABLE gestao_juros (
id_gestao_juros INT PRIMARY KEY AUTO_INCREMENT,
    juros_credito DECIMAL(10,2) NOT NULL,
    juros_emprestimo DECIMAL(10,2) NOT NULL
);
```

```
CREATE TABLE relatorio_mudan_juros(
id_rela_mod_juros INT PRIMARY KEY AUTO_INCREMENT,
id_gestao_juros INT NOT NULL,
tipo_juros VARCHAR(150) NOT NULL,
novo_juros DECIMAL(10.2) NOT NULL);
```

```
-- 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(13),
    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,
hora_transacao)
        VALUES(input_id_cliente_pf, input_valor_transacao,
input_tipo_transacao, input_id_beneficiario, CURDATE(),
CURTIME());

    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, hora_transacao)
        VALUES(input_id_cliente_pf, input_valor_transacao,
input_tipo_transacao, input_num_parcelas,
input_id_beneficiario, CURDATE(), CURTIME());

        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, status_pagamento ='Pendente'
                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;


    UPDATE cliente_pf
    SET saldo = saldo + input_valor_total_emprestimo
    WHERE id_cliente_pf = input_id_cliente_pf;


    UPDATE emprestimo_pf
    SET valor_pendente = valor_total_emprestimo,
        status_emprestimo = 'Pendente'
    WHERE id_emprestimo_pf = input_id_emprestimo;


    INSERT INTO lista_emprestimo_pf(id_emprestimo_pf,
num_parcela,
data_emprestimo,valor_pago,valor_total_emprestimo,
status_emprestimo )
    VALUES(input_id_emprestimo, input_quant_parcelas,
NOW(),0,valor_total_emprestimo, 'Pendente');

    SET id_emprestimo = LAST_INSERT_ID();


    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');


        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;
    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;


        UPDATE lista_emprestimo_pf
        SET valor_pago = valor_pago + input_valor_pago
        WHERE status_emprestimo = 'Pendente' AND
id_emprestimo_pf = (SELECT id_emprestimo_pf
                                    FROM emprestimo_pf
                                    WHERE
id_cliente_emprestimo_pf = input_id_cliente_pf);

        IF (SELECT valor_pago FROM lista_emprestimo_pf
            WHERE status_emprestimo = 'Pendente' AND
id_emprestimo_pf = (SELECT id_emprestimo_pf
                                        FROM emprestimo_pf
                                        WHERE
id_cliente_emprestimo_pf = input_id_cliente_pf)) =
           (SELECT valor_total_emprestimo FROM
lista_emprestimo_pf
            WHERE status_emprestimo = 'Pendente' AND
id_emprestimo_pf = (SELECT id_emprestimo_pf
                                        FROM emprestimo_pf
                                        WHERE
id_cliente_emprestimo_pf = input_id_cliente_pf)) THEN

            UPDATE lista_emprestimo_pf
            SET status_emprestimo = 'Pago'
            WHERE id_emprestimo_pf = (SELECT id_emprestimo_pf
                                        FROM emprestimo_pf
                                        WHERE
id_cliente_emprestimo_pf = input_id_cliente_pf);
        END IF;
```

```
        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, hora_transacao)
    VALUES(input_id_cliente_pf, input_valor_pago, 'Débito',
CURDATE(), CURTIME());


    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(13),
    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 ;
```

```
#Função
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 ;


DELIMITER $$

CREATE FUNCTION veric_cpf_exis(input_cpf VARCHAR(11))
RETURNS BOOLEAN
READS SQL DATA
BEGIN
    DECLARE cpf_existe BOOLEAN;

    SELECT EXISTS (
        SELECT 1
        FROM cliente_pf
        WHERE cpf = input_cpf
    ) INTO cpf_existe;

    RETURN cpf_existe;
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, CURDATE());
    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, CURDATE());
    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, CURDATE());
    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, CURDATE());
    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_autorizacao',
OLD.senha_de_autorizacao, NEW.senha_de_autorizacao,
CURDATE());
    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, CURDATE());
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, CURDATE());
    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, CURDATE());
    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, CURDATE());
    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, CURDATE());
    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, CURDATE());
    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, CURDATE());
    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, CURDATE());
    END IF;
END$$
```

```
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,
CURDATE());
END$$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER regis_modif_ges_juros
AFTER UPDATE
ON gestao_juros
FOR EACH ROW
BEGIN
     IF OLD.juros_credito <> NEW.juros_credito THEN
          INSERT INTO relatorio_mudan_juros (id_gestao_juros,
tipo_juros, novo_juros)
        VALUES (OLD.id_gestao_juros,'Crédito',
NEW.juros_credito);
    END IF;

     IF OLD.juros_emprestimo <> NEW.juros_emprestimo THEN
          INSERT INTO relatorio_mudan_juros (id_gestao_juros
,tipo_juros, novo_juros)
        VALUES (OLD.id_gestao_juros,'Empréstimo',
NEW.juros_emprestimo);
    END IF;
```

```
END $$

DELIMITER ;
```

```
#View:

CREATE VIEW extrato_comum AS
SELECT
    ex.id_pagador,
    cf1.nome_cliente_pf AS nome_pagador,
    ex.valor_transacao,
    ex.tipo_pagamento,
    ex.num_parcela_atual,
    ex.num_parcela_total,
    cf2.nome_cliente_pf AS nome_beneficiario,
    ex.data_transacao,
    ex.hora_transacao
FROM extrato_pf ex
JOIN cliente_pf cf1 ON ex.id_pagador = cf1.id_cliente_pf
JOIN cliente_pf cf2 ON ex.id_beneficiario = cf2.id_cliente_pf
ORDER BY ex.data_transacao DESC ;
```

```
CREATE VIEW lista_compra_crédito AS
SELECT DISTINCT
       cf1.id_cliente_pf,
        lp.valor_parcela_mes,
        lp.mes_parcela,
        lp.ano_parcela,
        lp.tipo_de_pagamento,
       lp.parcela_atual,
       cf2.nome_cliente_pf AS nome_beneficiario
FROM lista_parcelas_pf lp
```

```
JOIN extrato_pf ex ON lp.id_cliente_pf_lista_parcela =
ex.id_pagador
JOIN cliente_pf cf1 ON ex.id_pagador = cf1.id_cliente_pf
JOIN cliente_pf cf2 ON ex.id_beneficiario = cf2.id_cliente_pf
ORDER BY lp.ano_parcela, lp.mes_parcela;
```

```
CREATE VIEW veric_empres_atual AS
SELECT
    em.id_cliente_emprestimo_pf,
    le.valor_total_emprestimo,
    em.valor_pendente,
    le.data_emprestimo,
    pp.valor_total_mes,
    (SELECT COUNT(*)
     FROM pagamento_pendente_pf pp_sub
     WHERE pp_sub.id_lista_emprestimo =
le.id_lista_emprestimo
       AND pp_sub.status_pagamento = 'Pago') AS
num_parcelas_pagas,
    le.num_parcela AS num_parcela_total,
    pp.status_pagamento
FROM
    lista_emprestimo_pf le
JOIN
    emprestimo_pf em ON le.id_emprestimo_pf =
em.id_emprestimo_pf
JOIN
    pagamento_pendente_pf pp ON le.id_lista_emprestimo =
pp.id_lista_emprestimo
WHERE
    pp.status_pagamento = 'Pendente';
```

```
CREATE VIEW consul_empr_ger AS
SELECT em.id_cliente_emprestimo_pf,
```

```
           le.valor_total_emprestimo,
        le.num_parcela,
        le.data_emprestimo
FROM lista_emprestimo_pf le
JOIN emprestimo_pf em ON le.id_emprestimo_pf =
em.id_emprestimo_pf;
```

```
CREATE VIEW rela_modic_ender_clien AS
SELECT
cl.id_cliente_pf,
cl.nome_cliente_pf,
        mec.logradouro_antigo,
           mec.bairro_antigo,
           mec.cidade_antigo,
           mec.estado_antigo,
           mec.pais_antigo,
           mec.referencia_antigo,
           mec.data_modificacao
FROM modificar_endereco_cliente_pf mec
JOIN endereco_pf en ON mec.id_endereco_pf = en.id_endereco_pf
JOIN cliente_pf cl ON en.id_cliente = cl.id_cliente_pf;
```

```
CREATE VIEW rela_modic_ender_func AS
 SELECT
fu.id_funcionario,
fu.nome_funcionario,
 mef.logradouro_antigo,
 mef.bairro_antigo,
 mef.cidade_antigo,
 mef.estado_antigo,
 mef.pais_antigo,
 mef.referencia_antigo,
```

```
 mef.data_modificacao
 FROM modificar_endereco_funcionario mef
 JOIN endereco_funcionario enf ON mef.id_endereco_funcionario
= enf.id_endereco_funcionario
 JOIN funcionario fu ON enf.id_funcionario =
fu.id_funcionario;


CREATE VIEW situa_finac_clien AS
SELECT cl.id_cliente_pf,
           cl.nome_cliente_pf,
        cr.valor_disponivel AS val_disp_cred,
        cr.valor_total AS val_tot_cred,
        emp.valor_pendente AS val_pend_empr,
        emp.valor_total_disponivel as val_tot_empr,
        emp.status_emprestimo as status_empre
FROM cliente_pf cl
JOIN credito_pf cr ON cl.id_cliente_pf =
cr.id_cliente_credito_pf
JOIN emprestimo_pf emp ON cl.id_cliente_pf
=emp.id_cliente_emprestimo_pf;



CREATE VIEW veric_trans_alta AS
SELECT  cl1.id_cliente_pf,
           cl1.nome_cliente_pf AS Pagador,
        cl2.nome_cliente_pf AS Beneficiario,
           ex.valor_transacao,
           ex.tipo_pagamento,
        ex.data_transacao,
        ex.hora_transacao
FROM extrato_pf ex
JOIN cliente_pf cl1 ON ex.id_pagador = cl1.id_cliente_pf
JOIN cliente_pf cl2 ON ex.id_beneficiario = cl2.id_cliente_pf
WHERE valor_transacao > 10000 AND tipo_pagamento = 'Débito';


CREATE VIEW clien_alta_trans AS
SELECT
    cl.id_cliente_pf,
```

```
    cl.nome_cliente_pf,
    SUM(ex.valor_transacao) AS valor_total_trans
FROM extrato_pf ex
JOIN cliente_pf cl ON ex.id_pagador = cl.id_cliente_pf
GROUP BY cl.id_cliente_pf, cl.nome_cliente_pf
HAVING SUM(ex.valor_transacao) > 50000;
```

```
CREATE VIEW veric_parc_pend_empr AS
SELECT
     em.id_emprestimo_pf,
    pp.id_cliente_pf_pagamento,
    em.valor_pendente AS valor_total_emprestimo,
    pp.valor_total_mes AS valor_total_mes,
    pp.status_pagamento,
    pp.mes_parcela,
    pp.ano_parcela,
    em.status_emprestimo
FROM pagamento_pendente_pf pp
JOIN emprestimo_pf em ON pp.id_cliente_pf_pagamento =
em.id_cliente_emprestimo_pf
WHERE pp.tipo_de_pagamento = 'Empréstimo'
  AND em.status_emprestimo = 'Pendente'
  AND pp.status_pagamento = 'Pendente'
ORDER BY pp.id_lista_emprestimo, pp.ano_parcela,
pp.mes_parcela;
```