Dupla: Luís M. T. e Guilherme Bedin

-- Exercício 1

```
CREATE OR REPLACE FUNCTION sala_medio (nome VARCHAR) RETURN NUMBER
IS
   fno_func NUMBER;
   total_salario NUMBER := 0;
   qtde_registros NUMBER := 0;
   salario_medio NUMBER;
BEGIN
   -- Pega o FNO
   SELECT fno INTO fno_func
   FROM funcionario
   WHERE fnome = nome;

   -- Calcula o total de salários do funcionário
   SELECT SUM(salario) INTO total_salario
   FROM funcsal
   WHERE fno = fno_func;

   SELECT COUNT(*) INTO qtde_registros
   FROM funcsal
   WHERE fno = fno_func;

   -- Calcula o salário médio se houver registros
   IF qtde_registros > 0 THEN
      salario_medio := total_salario / qtde_registros;
   ELSE
      salario_medio := 0;
   END IF;

   -- Retorna o salário médio
   RETURN salario_medio;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 0;  -- Retorna 0 se não houver dados para o funcionário
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20001, 'Erro ao calcular o salário médio do
funcionário.');
END;
```

-- Exercício 2

```
CREATE OR REPLACE FUNCTION sala_medio_geren (data1 DATE, data2 DATE)
RETURN NUMBER
IS
   fno_geren NUMBER;
   total_salario NUMBER := 0;
   qtde_registros NUMBER := 0;
```

```
    salario_medio NUMBER;
BEGIN
    -- Pega o FNO
    SELECT fno INTO fno_geren
    FROM projeto_funcionario
    WHERE responsabilidade = 'Gerente';

    -- Calcula o total de salários dos gerentes
    SELECT SUM(salario) INTO total_salario
    FROM funcsal
    WHERE fno = fno_geren
    AND data BETWEEN data1 AND data2; -- Corrigido: data_salario deve ser o nome da
coluna

    -- Verifica a quantidade de registros encontrados
    SELECT COUNT(*) INTO qtde_registros
    FROM funcsal
    WHERE fno = fno_geren
    AND data BETWEEN data1 AND data2; -- Corrigido: data_salario deve ser o nome da
coluna

    -- Calcula o salário médio se houver registros
    IF qtde_registros > 0 THEN
        salario_medio := total_salario / qtde_registros;
    ELSE
        salario_medio := 0;
    END IF;

    -- Retorna o salário médio
    RETURN salario_medio;
END;

-- Exercício 3
create or replace procedure add_bonus as
begin
    update funcsal
    set salario = salario + 500
    where data < sysdate - INTERVAL '1' YEAR;

    commit;
end;

exec add_bonus;

-- Exercício 4
CREATE VIEW funcionarios AS
SELECT
    f.fnome AS nome,
```

```
    s.salario AS salario_atual,
    c.salario AS salario_base,
    s.salario - c.salario AS diferenca_salario,
    f.dataadm AS data_admissao,
    TRUNC(SYSDATE - f.dataadm) AS tempo_empresa
FROM
    funcionario f
JOIN
    funcsal s ON f.fno = s.fno
JOIN
    cargo c ON c.cod_cargo = f.cod_cargo;

-- Exercício 5
CREATE OR REPLACE TRIGGER funcsal_irpf
AFTER INSERT ON funcsal
FOR EACH ROW
DECLARE
    v_irpf NUMBER;
BEGIN
    -- Calcula o IRPF com base no salário inserido e nas alíquotas de 2019
    IF :NEW.salario <= 1903.98 THEN
        v_irpf := 0;
    ELSIF :NEW.salario <= 2826.65 THEN
        v_irpf := (:NEW.salario * 0.075) - 142.80;
    ELSIF :NEW.salario <= 3751.05 THEN
        v_irpf := (:NEW.salario * 0.15) - 354.80;
    ELSIF :NEW.salario <= 4664.68 THEN
        v_irpf := (:NEW.salario * 0.225) - 636.13;
    ELSE
        v_irpf := (:NEW.salario * 0.275) - 869.36;
    END IF;

    -- Insere o valor do IRPF calculado na tabela funcirpf
    INSERT INTO funcirpf (fno, irpf, data) VALUES (:NEW.fno, v_irpf, CURRENT_DATE());
END;

-- Exercício 6
CREATE OR REPLACE TRIGGER sal_receb_descont
AFTER INSERT ON funcirpf
FOR EACH ROW
DECLARE
    salario_func DECIMAL(10, 2);
    salario_descontado DECIMAL(10, 2);
BEGIN
    select salario into salario_func
    from funcsal
    where funcsal.fno = NEW.fno;
```

```
    salario_descontado := salario_func - NEW.irpf;

    -- Insere o valor do IRPF calculado na tabela funcirpf
    INSERT INTO salario_a_receber (fno, saldesconto, data) VALUES (:NEW.fno,
salario_descontado, CURRENT_DATE());
END;

-- Exercício 7
CREATE OR REPLACE TRIGGER salar_bonus
AFTER INSERT ON bonus
FOR EACH ROW
DECLARE
    salario_func DECIMAL(10, 2);
    ultima_atualizacao DATE;
BEGIN
    SELECT MAX(data) INTO ultima_atualizacao FROM funcsal;

    -- Adiciona o valor do bônus ao salário de todos os funcionários
    UPDATE funcsal
    SET salario = salario + NEW.valor
    WHERE data = ultima_atualizacao;
END;

-- Exercício 8
create or replace trigger mostrar_diferenca_salario
before update on funcsal
for each row
declare
    salario_antigo number;
begin
    salario_antigo := :OLD.salario;
    dbms_output.put_line('Salario Antigo: ' || salario_antigo || ' Novo Salario: ' || :NEW.salario
|| ' Diferença: ' || (:NEW.salario - salario_antigo));
end;

SET SERVEROUTPUT ON;

-- Exercício 9
create table funcsal_backup as select * from funcsal where 1=0;
drop table funcsal_backup;

select * from funcsal_backup;

-- Exercício 10
create or replace trigger backup_delete
after delete on funcsal
for each row
begin
```

```
    insert into funcsal_backup(fno, salario, data)
    values (OLD.fno, OLD.salario, OLD.data);
end;

delete from funcsal where fno = 1000;
```