

2)

```
CREATE TABLE alunos (  
    matricula INT PRIMARY KEY,  
    nome VARCHAR(100)  
);
```

```
CREATE TABLE cursos (  
    codigo INT PRIMARY KEY,  
    nome VARCHAR(100)  
);
```

```
CREATE TABLE matriculas (  
    codigo_mat INT PRIMARY KEY,  
    matricula INT,  
    codigo INT,  
    FOREIGN KEY (matricula) REFERENCES alunos(matricula),  
    FOREIGN KEY (codigo) REFERENCES cursos(codigo)  
);
```

```
INSERT INTO alunos (matricula, nome) VALUES  
(1, 'João'),  
(2, 'Maria'),  
(3, 'Pedro'),  
(4, 'Gabriela');
```

```
INSERT INTO cursos (codigo, nome) VALUES  
(101, 'Direito'),  
(102, 'Sistemas de Informação'),  
(103, 'Ontopsicologia');
```

```
INSERT INTO matriculas (codigo_mat, matricula, codigo) VALUES  
(1, 1, 101),  
(2, 2, 102),  
(3, 3, 101),  
(4, 4, 103);
```

```
SELECT  
    c.nome AS curso,  
    COUNT(m.matricula) AS total_alunos  
FROM  
    cursos c  
JOIN  
    matriculas m ON c.codigo = m.codigo  
JOIN  
    alunos a ON a.matricula = m.matricula  
GROUP BY  
    c.nome; -- Agrupa os resultados por curso
```

4)

CREATE OR REPLACE PROCEDURE FAZ_ALGO(X INTEGER, OUT Y REAL)

Isso cria ou substitui um procedimento chamado FAZ_ALGO

O parâmetro X é do tipo INTEGER

O parâmetro Y é uma saída (retorna o valor) e é do tipo REAL

AS \$\$ Define o início do bloco de código PL/pgSQL

DECLARE: onde variáveis podem ser declaradas, declara a variável subtotal que será um apelido (alias) para o primeiro parâmetro passado (\$1).

subtotal ALIAS FOR \$1; Neste caso, o subtotal deveria ser um alias para X, não \$1.

BEGIN onde as operações são executada

Y := subtotal + subtotal * 0.05; A variável de saída Y recebe o valor do subtotal (o parâmetro X com um alias), acrescido de 5%. Portanto, Y será X + 5% de X

END; Termina

\$\$ LANGUAGE plpgsql; Determina que linguagem será usada

SELECT FAZ_ALGO(100)

Esse é um erro, pois procedures não podem ser chamadas diretamente com SELECT. A maneira correta de chamar uma procedure seria utilizando a função CALL:

5) As Functions são conjuntos de códigos que sempre retornam um valor, como textos, datas ou números, enquanto as Procedures, também são um bloco de códigos, porém podem ou não retornar valores, são geralmente usados para executar uma série de comandos sem a necessidade de retorno de informações.

7) **CREATE OR REPLACE FUNCTION nome_funcao(NUMERIC)**

cria ou substitui uma função chamada nome_funcao, que recebe um parâmetro do tipo NUMERIC.

VARCHAR(20) AS \$\$

a função retorna um conjunto de strings do tipo VARCHAR(20).

DECLARE registro RECORD; x ALIAS FOR \$1;

define uma variável registro que armazena uma linha de dados da consulta SELECT. x é um alias para o parâmetro de entrada.

BEGIN inicia a lógica da função

FOR registro IN SELECT * FROM Empregado WHERE salario >= x

loop que itera sobre todas as linhas da tabela Empregado onde o salario (salário) é maior ou igual ao valor de x.

LOOP RETURN NEXT registro.nome;

END LOOP;

percorre os resultados da consulta e retorna o nome de cada empregado que atenda à condição. Cada vez que o RETURN NEXT é chamado, um novo resultado é retornado.

\$\$ LANGUAGE plpgsql; Determina que linguagem será usada

```
8) CREATE PROCEDURE AtualizarPrecoProduto(  
    IN cod_produto INT,  
    IN novo_preco DECIMAL(10, 2)  
)  
BEGIN  
    UPDATE produto  
    SET p_preco = novo_preco  
    WHERE p_cod_produto = cod_produto;  
  
    IF ROW_COUNT() = 0 THEN  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Produto não encontrado';  
    END IF;  
END;
```