



PL/SQL

Guilherme Lima Fontana 146896

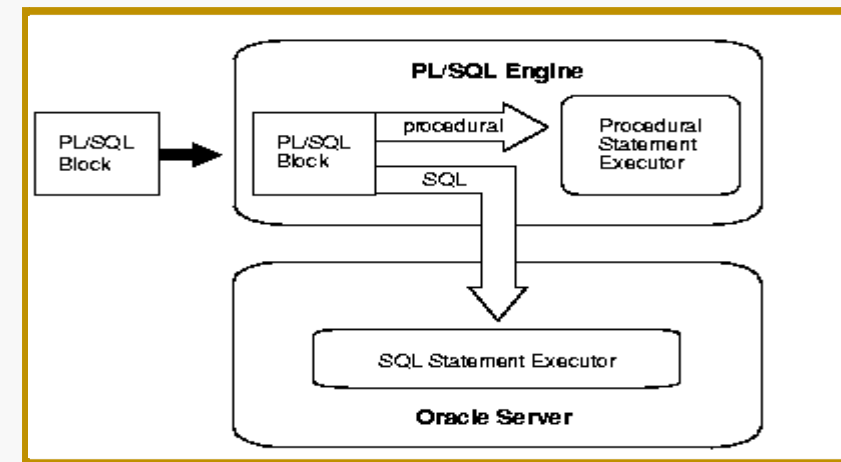
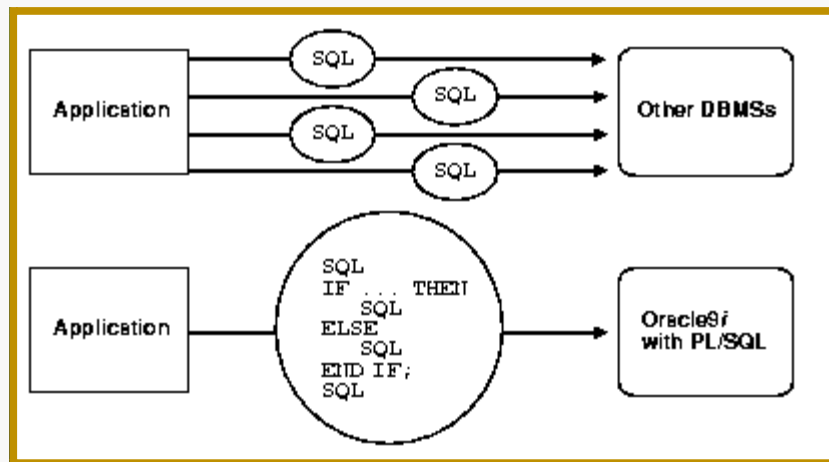
Tiago França Gurdiano 057800

Sávio Marinho da Silva 173564



Breve introdução ao PL/SQL...

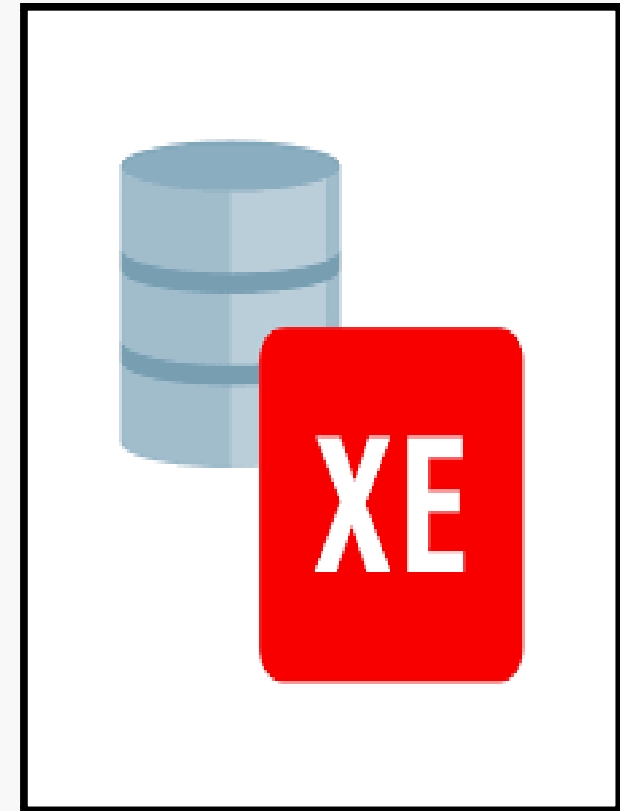
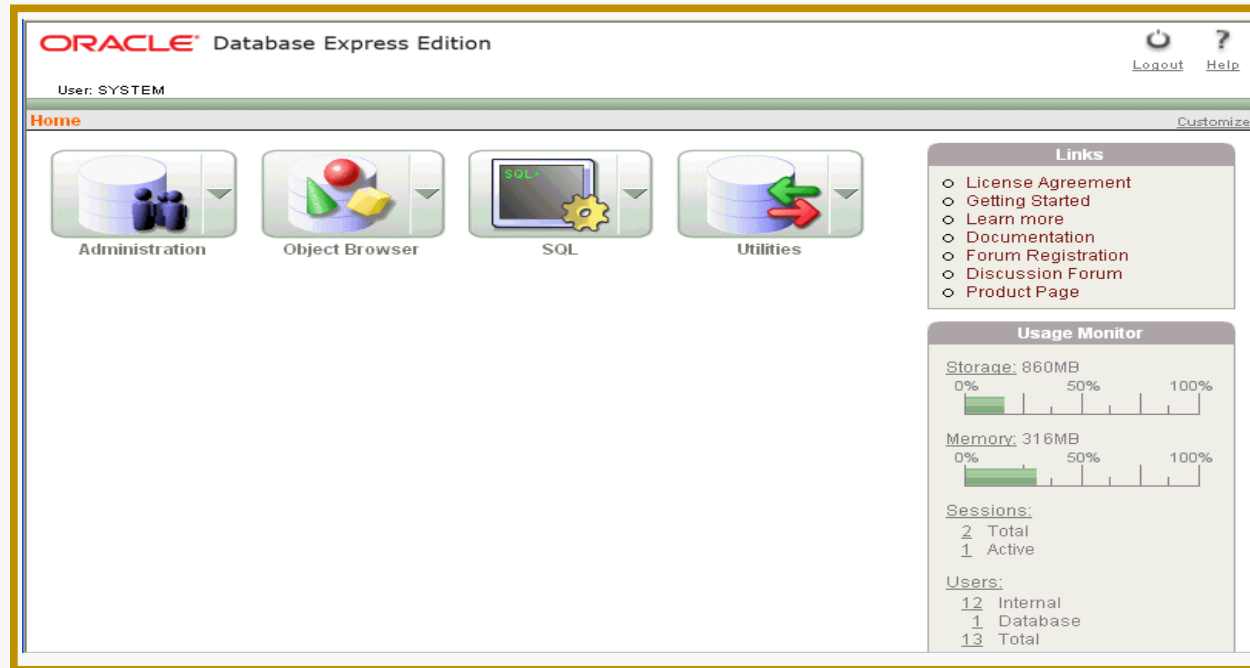
- PL/SQL é uma extensão procedural da linguagem SQL desenvolvida pela Oracle. É utilizada para escrever código no banco de dados Oracle, permitindo a criação de procedimentos armazenados, funções, gatilhos (triggers) e pacotes. PL/SQL combina a linguagem de consulta de dados SQL com elementos de programação procedural, oferecendo recursos como variáveis, estruturas de controle de fluxo e manipulação de exceções.



PL/SQL na prática.

- Para testar um código PL/SQL, você tem algumas opções, tanto para instalar no seu computador quanto para usar ambientes online:

Escolhendo o sistema de gerenciamento de banco de dados...



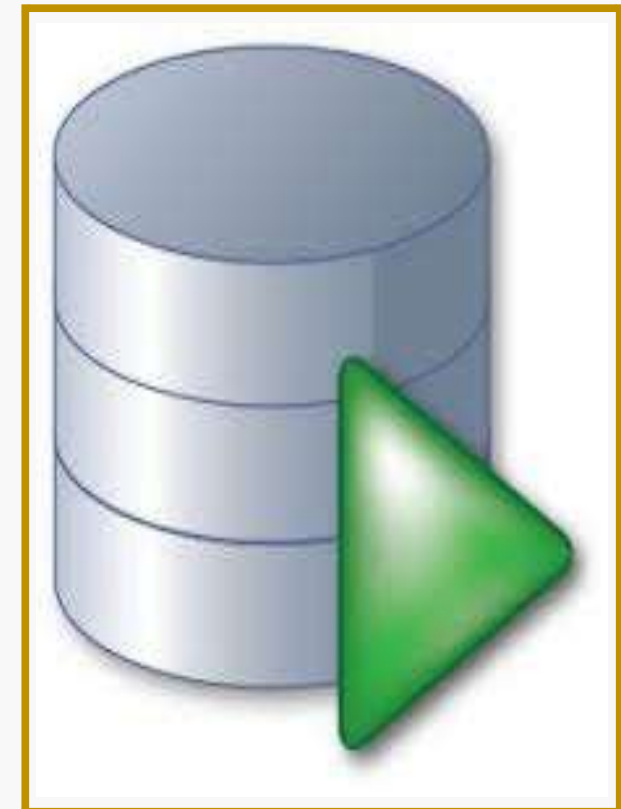
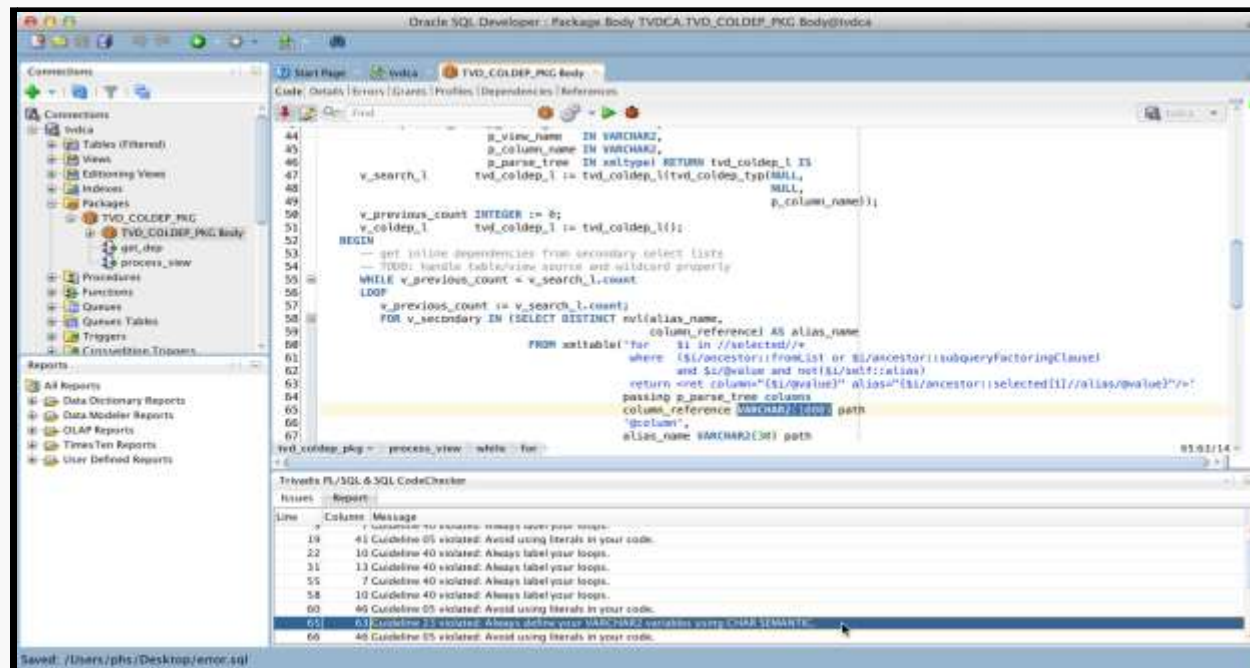
■ Oracle Database Express Edition (XE):

*É uma versão gratuita e mais leve do banco de dados Oracle. Você pode baixá-la e instalá-la no seu computador para testar o PL/SQL localmente. Após a instalação, você pode usar o SQL*Plus ou o Oracle SQL Developer para escrever e executar seu código PL/SQL.*

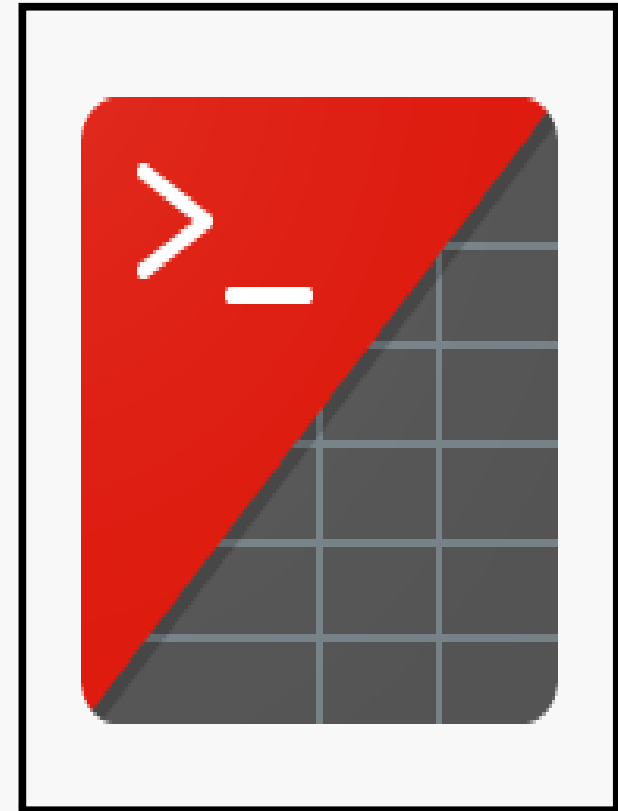
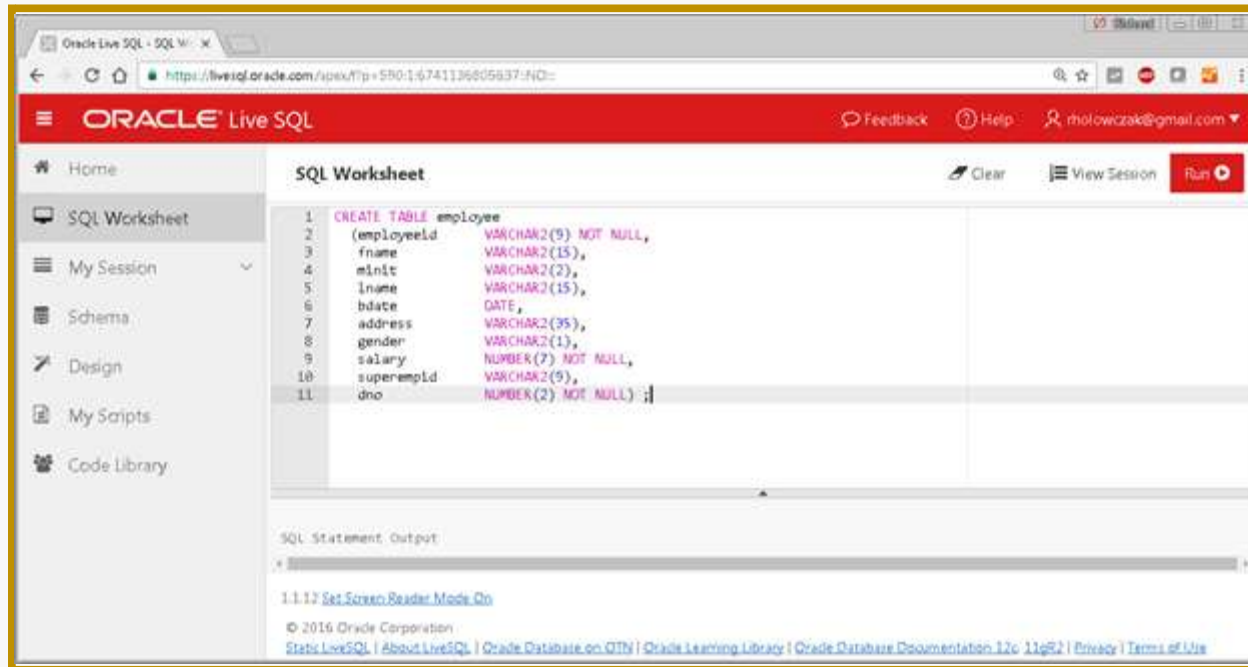
Escolhendo o sistema de gerenciamento de banco de dados...

■ Oracle SQL Developer:

É uma ferramenta gráfica gratuita que facilita o desenvolvimento e a gestão de bases de dados Oracle. Ela é útil para escrever e testar códigos PL/SQL. No entanto, ela requer um banco de dados Oracle para se conectar, então você ainda precisaria do Oracle XE ou de outro servidor Oracle.



Escolhendo o sistema de gerenciamento de banco de dados...



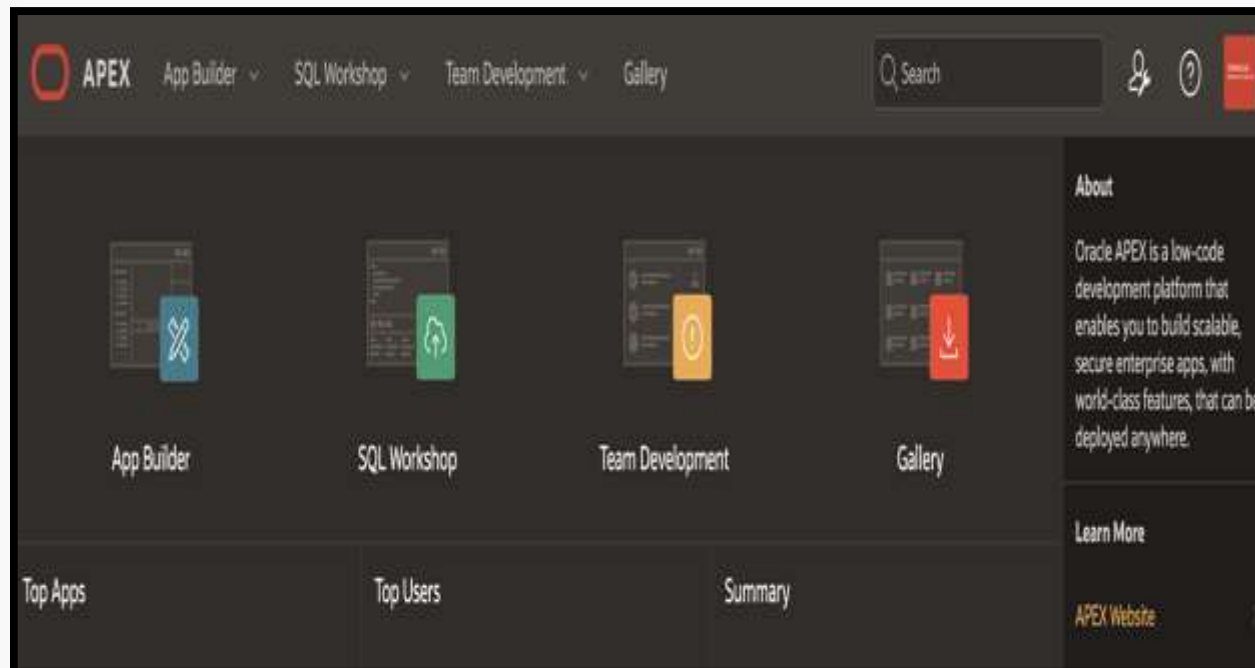
■ Oracle Live SQL:

É um serviço online gratuito fornecido pela Oracle que permite escrever, executar e compartilhar scripts SQL e PL/SQL. É uma ótima opção se você não quer instalar nada no seu computador. Você pode acessar o Oracle Live SQL em live.oracle.com.

Escolhendo o sistema de gerenciamento de banco de dados...


■ Apex Oracle:

Outra opção é o Oracle Application Express (APEX), que também oferece um ambiente online para trabalhar com SQL e PL/SQL. Acesse através de apex.oracle.com



Hello World no PL/SQL.

Um exemplo simples para começar...

 Exemplo

```
1  DECLARE
2  |      message VARCHAR2(20) := 'Hello, World!';
3  BEGIN
4  |      DBMS_OUTPUT.PUT_LINE(message);
5  END;
```

Este código PL/SQL define uma variável mensagem contendo a string 'Hello, World!' e em seguida a imprime na saída padrão usando DBMS_OUTPUT.PUT_LINE.

Trabalhando com Variáveis

Trabalhar com variáveis em PL/SQL envolve algumas etapas básicas:

- *Declaração de Variáveis:*

Você começa declarando suas variáveis na seção DECLARE de um bloco PL/SQL. Cada variável precisa de um tipo específico, como VARCHAR2 para strings, NUMBER para números, etc.

- *Inicialização de Variáveis:*

Opcionalmente, você pode inicializar variáveis com valores específicos na sua declaração.

- *Atribuição de Valores:*

Durante a execução do bloco PL/SQL, você pode atribuir ou alterar o valor das variáveis usando o operador de atribuição (:=").

Trabalhando com Variáveis

Aqui está um exemplo prático que inclui todos esses passos:

```
Exemplo
1  DECLARE
2      -- Declaração de variáveis
3      uma_string VARCHAR2(50);
4      um_numero  NUMBER;
5  BEGIN
6      -- Atribuição de valores
7      uma_string := 'Olá, PL/SQL!';
8      um_numero  := 100;
9
10     -- Usando as variáveis
11     DBMS_OUTPUT.PUT_LINE('String: ' || uma_string);
12     DBMS_OUTPUT.PUT_LINE('Número: ' || um_numero);
13 END;
```

Neste exemplo:

- `uma_string` é uma variável do tipo `VARCHAR2` que pode conter até 50 caracteres.
- `um_numero` é uma variável do tipo `NUMBER`.
- Os valores são atribuídos a essas variáveis na seção `BEGIN`.
- Finalmente, os valores são exibidos na saída padrão.

Controle de Fluxo: IF...ELSE e Loops.

- Estruturas de controle de fluxo, como IF...ELSE e loops (FOR, WHILE), são fundamentais em PL/SQL para controlar a execução do código com base em condições específicas...

IF...ELSE

A estrutura IF...ELSE é usada para executar blocos de código com base em determinadas condições.

```
Exemplo
1  DECLARE
2    num NUMBER := 10;
3  BEGIN
4    IF num > 5 THEN
5      DBMS_OUTPUT.PUT_LINE('O número é maior que 5');
6    ELSE
7      DBMS_OUTPUT.PUT_LINE('O número é 5 ou menor');
8    END IF;
9  END;
```

Neste exemplo:

- O programa verifica se a variável num é maior que 5. Se for, imprime uma mensagem; caso contrário, imprime outra mensagem.

FOR Loop

O loop FOR é usado para iterar sobre um intervalo de valores.

```
Exemplo
1  DECLARE
2    num NUMBER := 10;
3  BEGIN
4    IF num > 5 THEN
5      DBMS_OUTPUT.PUT_LINE('O número é maior que 5');
6    ELSE
7      DBMS_OUTPUT.PUT_LINE('O número é 5 ou menor');
8    END IF;
9  END;
```

Neste exemplo:

- o loop FOR itera de 1 a 5, imprimindo o valor atual de i a cada iteração.

WHILE Loop

O loop **WHILE** executa um bloco de código **repetidamente** enquanto uma **condição específica é verdadeira**.

Exemplo

```
1 DECLARE
2     contador NUMBER := 1;
3 BEGIN
4     WHILE contador <= 5 LOOP
5         DBMS_OUTPUT.PUT_LINE('Contador: ' || contador);
6         contador := contador + 1;
7     END LOOP;
8 END;
```

Neste exemplo:

- o loop **WHILE** continua executando enquanto contador for **menor ou igual a 5**, incrementando **contador** a cada iteração.

FUNCTION

Em PL/SQL, uma função é um tipo de bloco PL/SQL que permite que você **encapsule** uma **tarefa específica** e possa **retornar um valor**. Funções são úteis para realizar cálculos, processar informações e retornar um resultado.

Exemplo

```
1 CREATE OR REPLACE FUNCTION calcular_soma(p_numero1 NUMBER, p_numero2 NUMBER) RETURN NUMBER IS
2     resultado NUMBER;
3 BEGIN
4     resultado := p_numero1 + p_numero2;
5     RETURN resultado;
6 END calcular_soma;
```

Neste exemplo:

- **CREATE OR REPLACE FUNCTION calcular_soma**: Este comando cria uma nova função ou substitui uma existente chamada **calcular_soma**.
- **(p_numero1 NUMBER, p_numero2 NUMBER)**: São os **parâmetros** da função. **p_numero1** e **p_numero2** são dois números que serão somados.
- **RETURN NUMBER**: Indica que a função retornará um valor do tipo **NUMBER**.
- **resultado NUMBER**: Declara uma variável local chamada **resultado** para armazenar o resultado da soma.
- **BEGIN ... END**: Bloco de código que é executado quando a função é chamada. Ele calcula a soma de **p_numero1** e **p_numero2** e armazena o resultado na variável **resultado**.
- **RETURN resultado**: Retorna o valor da soma.

FUNCTION

Para usar esta função, você pode chamar a função `calcular_soma` dentro de um `bloco PL/SQL`, passando os valores necessários, como mostrado abaixo:

```
Exemplo
1  DECLARE
2      soma NUMBER;
3  BEGIN
4      soma := calcular_soma(5, 10);
5      DBMS_OUTPUT.PUT_LINE('A soma é: ' || soma);
6  END;
```

Neste exemplo:

- a função `calcular_soma` é chamada com os argumentos `5` e `10`, e o `resultado` é armazenado na `variável soma`, que é então impresso.

Referências...

- <https://dev.to/guis2286/uma-visao-sobre-plsql-3l8k>
- https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/01_oview.htm
- <https://www.salvis.com/blog/2014/04/30/trivadis-plsql-sql-codechecker-for-sql-developer-released/>
- <https://holowczak.com/getting-started-with-oracle-livesql/2/>
- <https://apex.oracle.com/pt-br/>