```
/*
### *1. IF*

A estrutura IF no T-SQL é usada para executar um bloco de código
  condicionalmente, com base na avaliação de uma expressão booleana. Você pode
  também incluir uma cláusula ELSE para especificar o que deve acontecer se a
  condição for falsa.

#### *Sintaxe*:

IF <condição>
BEGIN
    -- Código a ser executado se a condição for verdadeira
END
ELSE
BEGIN
    -- Código a ser executado se a condição for falsa
END
*/

--#### *Exemplo*:
DECLARE @Valor INT = 10;

IF @Valor > 5
BEGIN
    PRINT 'O valor é maior que 5';
END
ELSE
BEGIN
    PRINT 'O valor é 5 ou menor';
END


/*
### *2. CASE (Equivalente ao SWITCH)*

O CASE é usado para executar uma expressão ou bloco de código com base em
  diferentes condições. Ele pode ser usado tanto em consultas para retornar
  valores baseados em condições quanto em blocos de código.

#### *Sintaxe*:
SELECT
    CASE
        WHEN <condição1> THEN <resultado1>
        WHEN <condição2> THEN <resultado2>
        ELSE <resultado_default>
    END AS Resultado
FROM
    <tabela>;
*/

--#### *Exemplo*:
DECLARE @DiaSemana INT = 3;
```

```
SELECT
    CASE @DiaSemana
        WHEN 1 THEN 'Domingo'
        WHEN 2 THEN 'Segunda-feira'
        WHEN 3 THEN 'Terça-feira'
        ELSE 'Outro dia'
    END AS NomeDia;

/*
### *3. WHILE*

A estrutura WHILE no T-SQL é usada para repetir um bloco de código enquanto uma
  condição específica for verdadeira. É semelhante ao loop WHILE em outras
  linguagens de programação.

#### *Sintaxe*:
WHILE <condição>
BEGIN
    -- Código a ser executado enquanto a condição for verdadeira
END
*/

--#### *Exemplo*:
DECLARE @Contador INT = 1;

WHILE @Contador <= 5
BEGIN
    PRINT 'Contagem: ' + CAST(@Contador AS VARCHAR(10));
    SET @Contador = @Contador + 1;
END;

/*
### *4. TRY-CATCH*

O bloco TRY-CATCH no T-SQL é usado para capturar e tratar exceções ou erros que
  possam ocorrer durante a execução de um bloco de código. Se um erro ocorrer no
   bloco TRY, o controle é transferido para o bloco CATCH, onde você pode lidar
  com o erro.

#### *Sintaxe*:
BEGIN TRY
    -- Código que pode causar um erro
END TRY
BEGIN CATCH
    -- Código para lidar com o erro
    -- Funções como ERROR_MESSAGE() podem ser usadas aqui
END CATCH
*/

--#### *Exemplo*:
BEGIN TRY
    -- Tentativa de divisão por zero, que causará um erro
```

```
    DECLARE @Resultado INT;
    SET @Resultado = 10 / 0;
END TRY
BEGIN CATCH
    PRINT 'Ocorreu um erro: ' + ERROR_MESSAGE();
END CATCH;

-- ### Bônus: Lançamento de erro.
/*
A função RAISERROR no T-SQL é utilizada para gerar mensagens de erro personalizadas e retornar essas mensagens ao chamador, seja ele uma aplicação cliente ou outro procedimento T-SQL. Essa função é frequentemente usada dentro de blocos de controle de erro, como TRY-CATCH, para relatar erros específicos com mensagens e níveis de severidade customizados.

### *Sintaxe*


RAISERROR (message_string, severity, state [, argument [, ...]])


- *message_string*: A mensagem de erro que será exibida. Pode incluir marcadores de substituição (%s, %d, etc.) para valores que serão passados como argumentos.
- *severity*: Um número inteiro entre 0 e 25 que indica a gravidade do erro. Valores comuns são:
  - *0-10*: Mensagens informativas.
  - *11-16*: Erros que indicam problemas no código ou lógica do usuário.
  - *17-25*: Erros mais graves, geralmente relacionados ao ambiente ou à integridade do banco de dados.
- *state*: Um número inteiro entre 0 e 255 que indica a origem do erro. O valor é geralmente 1.
- *argument*: Valores que serão substituídos nos marcadores da message_string.
*/
--### *Exemplos de Uso*

--#### *1. Simples RAISERROR*

RAISERROR('MENSAGEM PADRAO', 16, 1);

/*
#### *2. RAISERROR com Substituição de Argumentos*

Você pode usar substituições na mensagem para incluir valores dinâmicos:
*/
DECLARE @ErroMensagem NVARCHAR(255) = 'Ocorreu um erro no produto %s com o ID %d.';
DECLARE @Produto NVARCHAR(50) = 'Notebook';
DECLARE @ProdutoID INT = 1001;

RAISERROR(@ErroMensagem, 16, 1, @Produto, @ProdutoID);

/*
```

```
#### *3. Uso com TRY-CATCH*

O RAISERROR é frequentemente utilizado dentro de blocos TRY-CATCH para relatar
  erros personalizados:

*/
BEGIN TRY
    -- Simulação de um erro
    DECLARE @Divisor INT = 0;
    DECLARE @Resultado INT;

    SET @Resultado = 10 / @Divisor;
END TRY
BEGIN CATCH
    RAISERROR ('Divisão por zero não é permitida.', 16, 1);
END CATCH;


-- Neste exemplo, se ocorrer uma divisão por zero, a mensagem personalizada será
   gerada pelo RAISERROR.
```