Universidade Presbiteriana Mackenzie
Banco de Dados
Profa. Elisângela Botelho Gracias

GABARITO

Exercícios da Aula 19 – Linguagem PL/SQL

1) Desenvolva um bloco PL/SQL que faça a **atualização do salário do funcionário de código 103**, **observando os seguintes critérios**:

- Se o salário deste funcionário de código 103 for menor que 1500, então aumente o salário dele em 30%;
- Senão, se o salário dele for menor ou igual a 2000, aumente em 20%;
- Senão, aumente somente em 10%;
- No final, mostre na tela o valor do novo salário do funcionário 103.

```
SET SERVEROUTPUT ON;

DECLARE
      valor_salario int := 0;
BEGIN
    SELECT Salario INTO valor_salario
    FROM Funcionario
    WHERE (Cod_Func = 103);

    IF (valor_salario < 1500) THEN
            UPDATE Funcionario
            SET Salario = Salario * 1.3
            WHERE (Cod_Func = 103);
    ELSIF (valor_salario < 2000) THEN
            UPDATE Funcionario
            SET Salario = Salario * 1.2
            WHERE (Cod_Func = 103);
    ELSE
            UPDATE Funcionario
            SET Salario = Salario * 1.1
            WHERE (Cod_Func = 103);
    END IF;

    SELECT Salario INTO valor_salario
    FROM Funcionario
    WHERE (Cod_Func = 103);

    Dbms_Output.Put_Line('Novo salário: '|| valor_salario);
END;
```

2) **(utilize a estrutura WHILE)** Desenvolva um bloco PL/SQL que **aumente o salário do funcionário de código 101, de cada vez, em 10%, enquanto o salário dele for menor ou igual a 10000.**
A cada vez que o salário do funcionário de código 101 for aumentado, insira uma nova linha na tabela Historico_Salario (cujo script de criação se encontra logo abaixo).
No final, **mostre na tela o valor final do novo salário do funcionário 101**, após todas as atualizações feitas.

```
CREATE TABLE Historico_Salario
(Cod_Func INTEGER,
 Salario_Novo INTEGER,
 Dt_Atualizacao DATE,
 PRIMARY KEY(Cod_Func, Salario_Novo)
);
```

```
SET SERVEROUTPUT ON;

DECLARE
      valor_salario int := 0;
BEGIN
    SELECT Salario INTO valor_salario
    FROM Funcionario
    WHERE (Cod_Func = 101);

    WHILE (valor_salario <= 10000) LOOP
       UPDATE Funcionario
       SET Salario = Salario * 1.1
       WHERE (Cod_Func = 101);

       SELECT Salario INTO valor_salario
       FROM Funcionario
       WHERE (Cod_Func = 101);

       INSERT
       INTO Historico_Salario (Cod_Func, Salario_Novo, Dt_Atualizacao)
       VALUES (101, valor_salario, sysdate);

    END LOOP;

    Dbms_Output.Put_Line('Novo salário é: '|| valor_salario);

END;
```

3) Desenvolva um bloco PL/SQL que **mostre na tela** o **quanto o funcionário de nome 'Mario Souza' irá receber pelo número de horas que ele trabalhou nos projetos**, considerando que **cada hora trabalhada dele é 100**.

Mostre na tela, também, a **classificação deste funcionário**, de acordo com o número total de horas que ele trabalhou nos projetos, obedecendo aos seguintes critérios:

- se o número total de horas for maior ou igual a 200, então a classificação será "ótima participação";
- senão, se for maior ou igual a 100, então sua classificação será "boa participação";
- senão, sua classificação será "participação normal".

```
SET SERVEROUTPUT ON;

DECLARE
      valor INTEGER;
      horas INTEGER;

BEGIN
      SELECT SUM(FP.Horas_Trab) INTO horas
      FROM Funcionario F INNER JOIN Func_Proj FP
      ON (F.Cod_Func = FP.Cod_Func)
      WHERE (F.Nome_Func = 'Mario Souza');

      valor := horas*100;

      Dbms_Output.Put_Line('Valor a receber: '|| valor);


      IF (horas >= 200) THEN
            Dbms_Output.Put_Line('Ótima Participação');
      ELSIF (horas >= 100) THEN
            Dbms_Output.Put_Line('Boa Participação');
      ELSE
            Dbms_Output.Put_Line('Participação Normal');
      END IF;
END;
```