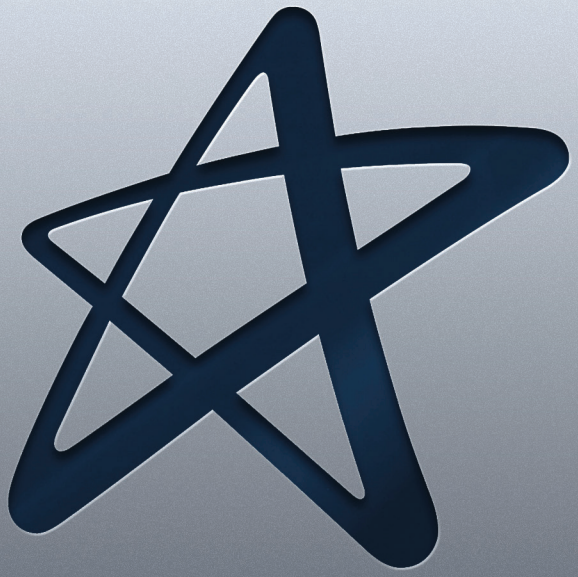
# Banco de Dados



Cruzeiro do Sul Virtual
Educação a distância

# Material Teórico

## Subquerys e Funções de Decisões

**Responsável pelo Conteúdo:**
Prof. Ms. Alexander Gobbato Albuquerque

**Revisão Textual:**
Profa. Esp.Vera Lídia de Sá Cicarone

# UNIDADE Subquerys e Funções de Decisões

- Conceito
- Operadores de comparação para múltiplas linhas
- Funções de Decisão

## Objetivo de APRENDIZADO

- Nessa fase estaremos utilizando vários conceitos aprendidos das outras unidades e acrescentando a esse conhecimento veremos as funções de decisões, essas funções de decisões podem nos auxiliar na formatação dos resultados retornados que irão facilitar em muito o nosso trabalho.

Lembramos a você a importância de realizar todas as atividades propostas dentro do prazo estabelecido para cada unidade. Dessa forma, evitará que o conteúdo se acumule e que você tenha problemas ao final do semestre.

Uma última recomendação: caso tenha problemas para acessar algum item da disciplina ou dúvidas com relação ao conteúdo, não deixe de entrar em contato com seu professor tutor através do botão mensagens.

# Contextualização

Com o conhecimento adquirido nos comandos DDL e DML, aprendemos a criar e manipular dados e com esse conhecimento adquirido podemos nos aprofundar um pouco mais nos conceitos de retornos de dados. Veremos nessa unidade que trabalhar com subquery é muito importante porque iremos combinar vários selects para obter um único resultado, esse resultado ainda pode vir acompanhado de uma função de decisão para formatar o resultado ou até mesmo contornar uma possível situação de falha.

Então está na hora de aprendermos a utilizar as subquerys e usufruir das funções de decisões.

# Conceito

Subquery é um comando select dentro de um outro comando select que retorna uma ou mais linhas, a fim de satisfazer uma cláusula WHERE.

A subquery (inner query), geralmente, é executada antes da consulta principal.

O resultado da subquery é, então, avaliado pelo da query principal (outer query).

Uma subpesquisa (subquery) é uma declaração SELECT que é aninhada com outra declaração SELECT, a qual retorna resultados intermediários.

A subquery é executada primeiro.

O resultado da subquery é utilizado na query principal.

Estrutura da subquery:

```
Select select_list

from table

Where expr operator

(Select select_list from table);
```

**Exemplo:**

Query principal

- Quais funcionários possuem um salário maior que o de Abel?
- Subquery
    - Qual o salário de Abel

```
SELECT last_name, salary

FROM HR.”EMPLOYEES”

Where salary > (SELECT salary

FROM HR.”EMPLOYEES”

WHERE last_name = ‘Abel’)
```

| LAST_NAME | SALARY |
|---|---|
| King | 24000 |
| Kochhar | 17000 |
| De Haan | 17000 |
| Greenberg | 12008 |
| Russell | 14000 |
| Partners | 13500 |
| Errazuriz | 12000 |
| Ozer | 11500 |
| Hartstein | 13000 |
| Higgins | 12008 |

**10 rows selected**

**Exemplo:**

Mostre o sobrenome (last_name) e o cargo (job_id) dos funcionários que possuem o mesmo cargo do funcionário de número 141.

```
SELECT last_name, job_id
FROM HR.”EMPLOYEES”
Where job_id = (SELECT job_id
FROM HR.”EMPLOYEES”
WHERE employee_id =141);
```

| LAST_NAME | JOB_ID |
|---|---|
| Nayer | ST_CLERK |
| Mikkilineni | ST_CLERK |
| Landry | ST_CLERK |
| Markle | ST_CLERK |
| Bissot | ST_CLERK |
| Atkinson | ST_CLERK |
| Marlow | ST_CLERK |
| Olson | ST_CLERK |
| Mallin | ST_CLERK |
| Rogers | ST_CLERK |
| Gee | ST_CLERK |
| Philtanker | ST_CLERK |
| Ladwig | ST_CLERK |
| Stiles | ST_CLERK |
| Seo | ST_CLERK |
| Patel | ST_CLERK |
| Rajs | ST_CLERK |
| Davies | ST_CLERK |
| Matos | ST_CLERK |
| Vargas | ST_CLERK |

**20 rows selected**

**Exemplo:**

Para encontrar todos os empregados que têm o mesmo cargo do David Lee, fazemos:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID
FROM EMPLOYEES
WHERE JOB_ID = (SELECT JOB_ID
FROM EMPLOYEES
         WHERE FIRST_NAME = 'David' and
LAST_NAME = 'Lee')
ORDER BY FIRST_NAME
```

```
(SELECT job_id
FROM employees
WHERE employee_id = 141);
```

| FIRST_NAME | LAST_NAME | JOB_ID |
|---|---|---|
| Allan | McEwen | SA_REP |
| Alyssa | Hutton | SA_REP |
| Amit | Banda | SA_REP |
| Charles | Johnson | SA_REP |
| Christopher | Olsen | SA_REP |
| Clara | Vishney | SA_REP |
| Danielle | Greene | SA_REP |
| David | Lee | SA_REP |
| David | Bernstein | SA_REP |
| Elizabeth | Bates | SA_REP |
| Ellen | Abel | SA_REP |
| Harrison | Bloom | SA_REP |
| Jack | Livingston | SA_REP |
| Janette | King | SA_REP |
| Jonathon | Taylor | SA_REP |
| Kimberely | Grant | SA_REP |
| Lindsey | Smith | SA_REP |
| Lisa | Ozer | SA_REP |
| Louise | Doran | SA_REP |
| Mattea | Marvins | SA_REP |
| Nanette | Cambrault | SA_REP |
| Oliver | Tuvault | SA_REP |
| Patrick | Sully | SA_REP |
| Peter | Tucker | SA_REP |
| Peter | Hall | SA_REP |
| Sarath | Sewall | SA_REP |
| Sundar | Ande | SA_REP |
| Sundita | Kumar | SA_REP |
| Tayler | Fox | SA_REP |
| William | Smith | SA_REP |

**30 rows selected**

**Exemplo:**

Para encontrar todos os empregados do mesmo departamento do David Lee, fazemos:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID, DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID = (  SELECT DEPARTMENT_ID
         FROM EMPLOYEES
         WHERE FIRST_NAME = 'David'
and LAST_NAME = 'Lee')
ORDER BY FIRST_NAME
```

| FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID |
|---|---|---|---|
| Alberto | Errazuriz | SA_MAN | 80 |
| Allan | McEwen | SA_REP | 80 |
| Alyssa | Hutton | SA_REP | 80 |
| Amit | Banda | SA_REP | 80 |
| Charles | Johnson | SA_REP | 80 |
| Christopher | Olsen | SA_REP | 80 |
| Clara | Vishney | SA_REP | 80 |
| Danielle | Greene | SA_REP | 80 |
| David | Bernstein | SA_REP | 80 |
| David | Lee | SA_REP | 80 |
| Eleni | Zlotkey | SA_MAN | 80 |
| Elizabeth | Bates | SA_REP | 80 |
| Ellen | Abel | SA_REP | 80 |
| Gerald | Cambrault | SA_MAN | 80 |
| Harrison | Bloom | SA_REP | 80 |
| Jack | Livingston | SA_REP | 80 |
| Janette | King | SA_REP | 80 |
| John | Russell | SA_MAN | 80 |
| Jonathon | Taylor | SA_REP | 80 |
| Karen | Partners | SA_MAN | 80 |
| Lindsey | Smith | SA_REP | 80 |
| Lisa | Ozer | SA_REP | 80 |
| Louise | Doran | SA_REP | 80 |
| Mattea | Marvins | SA_REP | 80 |
| Nanette | Cambrault | SA_REP | 80 |
| Oliver | Tuvault | SA_REP | 80 |
| Patrick | Sully | SA_REP | 80 |
| Peter | Hall | SA_REP | 80 |
| Peter | Tucker | SA_REP | 80 |
| Sarath | Sewall | SA_REP | 80 |
| Sundar | Ande | SA_REP | 80 |
| Sundita | Kumar | SA_REP | 80 |
| Tayler | Fox | SA_REP | 80 |
| William | Smith | SA_REP | 80 |

**34 rows selected**

A seguinte pesquisa busca encontrar os empregados que ganham acima da média salarial.

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID,
        DEPARTMENT_ID, SALARY
FROM EMPLOYEES
WHERE SALARY >  (SELECT AVG(SALARY)
        FROM EMPLOYEES)
ORDER BY FIRST_NAME
```

A pesquisa interna retorna o cargo de David Lee (SA_REP) o qual é usado na condição WHERE da pesquisa principal.

| FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID | SALARY |
|---|---|---|---|---|
| Adam | Fripp | ST_MAN | 50 | 8200 |
| Alberto | Errazuriz | SA_MAN | 80 | 12000 |
| Alexander | Hunold | IT_PROG | 60 | 9000 |
| Allan | McEwen | SA_REP | 80 | 9000 |
| Alyssa | Hutton | SA_REP | 80 | 8800 |
| Christopher | Olsen | SA_REP | 80 | 8000 |
| Clara | Vishney | SA_REP | 80 | 10500 |
| Daniel | Faviet | FI_ACCOUNT | 100 | 9000 |
| Danielle | Greene | SA_REP | 80 | 9500 |
| David | Lee | SA_REP | 80 | 6800 |
| David | Bernstein | SA_REP | 80 | 9500 |
| Den | Raphaely | PU_MAN | 30 | 11000 |
| Eleni | Zlotkey | SA_MAN | 80 | 10500 |
| Elizabeth | Bates | SA_REP | 80 | 7300 |
| Ellen | Abel | SA_REP | 80 | 11000 |
| Gerald | Cambrault | SA_MAN | 80 | 11000 |
| Harrison | Bloom | SA_REP | 80 | 10000 |
| Hermann | Baer | PR_REP | 70 | 10000 |
| Ismael | Sciarra | FI_ACCOUNT | 100 | 7700 |
| Jack | Livingston | SA_REP | 80 | 8400 |
| Janette | King | SA_REP | 80 | 10000 |
| John | Russell | SA_MAN | 80 | 14000 |
| John | Chen | FI_ACCOUNT | 100 | 8200 |
| Jonathon | Taylor | SA_REP | 80 | 8600 |
| Jose Manuel | Urman | FI_ACCOUNT | 100 | 7800 |
| Karen | Partners | SA_MAN | 80 | 13500 |
| Kimberely | Grant | SA_REP | - | 7000 |
| Lex | De Haan | AD_VP | 90 | 17000 |
| Lindsey | Smith | SA_REP | 80 | 8000 |
| Lisa | Ozer | SA_REP | 80 | 11500 |
| Louise | Doran | SA_REP | 80 | 7500 |
| Luis | Popp | FI_ACCOUNT | 100 | 6900 |
| Mattea | Marvins | SA_REP | 80 | 7200 |
| Matthew | Weiss | ST_MAN | 50 | 8000 |
| Michael | Hartstein | MK_MAN | 20 | 13000 |
| Nancy | Greenberg | FI_MGR | 100 | 12008 |
| Nanette | Cambrault | SA_REP | 80 | 7500 |
| Neena | Kochhar | AD_VP | 90 | 17000 |
| Oliver | Tuvault | SA_REP | 80 | 7000 |
| Patrick | Sully | SA_REP | 80 | 9500 |
| Payam | Kaufling | ST_MAN | 50 | 7900 |
| Peter | Tucker | SA_REP | 80 | 10000 |
| Peter | Hall | SA_REP | 80 | 9000 |
| Sarath | Sewall | SA_REP | 80 | 7000 |
| Shanta | Vollman | ST_MAN | 50 | 6500 |
| Shelley | Higgins | AC_MGR | 110 | 12008 |
| Steven | King | AD_PRES | 90 | 24000 |
| Susan | Mavris | HR_REP | 40 | 6500 |
| Tayler | Fox | SA_REP | 80 | 9600 |
| William | Smith | SA_REP | 80 | 7400 |

**50 rows selected**

A seguinte pesquisa busca encontrar os empregados que trabalham nos departamentos de Marketing e Administração.

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID,
    DEPARTMENT_ID, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN  (SELECT DEPARTMENT_ID
         FROM DEPARTMENTS
         WHERE DEPARTMENT_NAME IN
         ('Marketing','Administration'))
ORDER BY FIRST_NAME
```

SingleRowSubquery – Retorno de **um único valor** na subquerie

| FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID | SALARY |
|---|---|---|---|---|
| Jennifer | Whalen | AD_ASST | 10 | 4400 |
| Michael | Hartstein | MK_MAN | 20 | 13000 |
| Pat | Fay | MK_REP | 20 | 6000 |

**3 rows selected**

## Subpesquisas Aninhadas

Mostrar o primeiro e o último nome, cargo, código do departamento e salário dos empregados cujo salário é maior que o maior salário no departamento 'SALES'.

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID,
    DEPARTMENT_ID, SALARY
FROM EMPLOYEES
WHERE SALARY > ( SELECT MAX(SALARY)
         FROM EMPLOYEES
         WHERE DEPARTMENT_ID = (SELECT DEPARTMENT_ID
                   FROM DEPARTMENTS
                   WHERE DEPARTMENT_NAME = 'Sales'))
ORDER BY FIRST_NAME
```

| FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID | SALARY |
|---|---|---|---|---|
| Lex | De Haan | AD_VP | 90 | 17000 |
| Neena | Kochhar | AD_VP | 90 | 17000 |
| Steven | King | AD_PRES | 90 | 24000 |

**3 rows selected**

# Operadores de comparação para múltiplas linhas

| Operator | Significado |
|---|---|
| IN | Igual a qualquer membro da lista |
| ANY | Compara o valor com cada valor retornado pela subquerie. |
| ALL | Compara o valor com todos os valores retornados pela subqueire. |
| EXISTS | Testa se um valor existe. |

## Operador ANY

Compara o valor com cada valor retornado pela subquerie.

< ANY = menor que o maior valor

> ANY = maior que o menor valor

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|---|---|---|---|
| 124 | Mourgous | ST_MAN | 5800 |
| 141 | Rajs | ST_CLERK | 3500 |
| 142 | Davies | ST_CLERK | 3100 |
| 143 | Matos | ST_CLERK | 2600 |
| 144 | Vargas | ST_CLERK | 2500 |

## Operador ALL

Compara o valor com todos os valores retornados pela subquerie.

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|---|---|---|---|
| 141 | Rajs | ST_CLERK | 3500 |
| 142 | Davies | ST_CLERK | 3100 |
| 143 | Matos | ST_CLERK | 2600 |
| 144 | Vargas | ST_CLERK | 2500 |

# Funções de Decisão

## DECODE

Ele facilita pesquisas condicionais, fazendo o trabalho de 'ferramentas' ou comandos 'se-então-se não'.

Sintaxe:

DECODE(col/expressão,procurado1,resultado1,...,padrão)

Col/expressão é comparado com cada um dos valores procurados e separados por vírgula e retorna o resultado se a col/expressão for igual ao valor procurado.

Se não for encontrado nenhum dos valores procurados, a função DECODE retorna o valor padrão. Se o valor padrão for omitido, ele retornará um valor nulo.

### DECODE – Argumentos

DECODE deve ter, no mínimo, 3 parâmetros ou argumentos.

- √ COL/EXPRESSÃO – o nome da coluna ou expressão a ser avaliado.
- √ PROCURADO1 – o primeiro valor para ser testado.
- √ RESULTADO1 – o valor para ser retornado, se o procurado1 for encontrado.
- √ PROCURADO1 e RESULTADO1 podem ser repetidos quantas vezes forem necessárias. (PROCURADO2,RESULTADO2, PROCURADO3,RESULTADO3,...)
- √ PADRÃO – o valor a ser retornado, se nenhum procurado for encontrado.

***Nota:***

col/expressão pode ser vários tipos de dados.

PROCURADO deve ser um dado do tipo coluna ou expressão.

O valor retornado é forçado para alguns tipos de dados como o terceiro argumento (resultado1).

O seguinte exemplo decodifica os cargos dos tipos IT_PROG e CLERK unicamente. Os outros cargos serão padrão e alterados para INDEFINIDO.

```
SELECT FIRST_NAME, JOB_ID,
DECODE( JOB_ID,'IT_PROG','PROGRAMADOR',
         'FI_ACCOUNT','CONTADOR',
         'INDEFINIDO') DECODE_CARGO
FROM EMPLOYEES;
```

| FIRST_NAME | JOB_ID | DECODE_CARGO |
|---|---|---|
| Ellen | SA_REP | INDEFINIDO |
| Sundar | SA_REP | INDEFINIDO |
| Mozhe | ST_CLERK | INDEFINIDO |
| David | IT_PROG | PROGRAMADOR |
| Hermann | PR_REP | INDEFINIDO |
| Shelli | PU_CLERK | INDEFINIDO |
| Amit | SA_REP | INDEFINIDO |
| Elizabeth | SA_REP | INDEFINIDO |
| Sarah | SH_CLERK | INDEFINIDO |
| David | SA_REP | INDEFINIDO |
| Laura | ST_CLERK | INDEFINIDO |
| Harrison | SA_REP | INDEFINIDO |
| Alexis | SH_CLERK | INDEFINIDO |
| Anthony | SH_CLERK | INDEFINIDO |
| Gerald | SA_MAN | INDEFINIDO |
| Nanette | SA_REP | INDEFINIDO |
| John | FI_ACCOUNT | CONTADOR |
| Kelly | SH_CLERK | INDEFINIDO |
| Karen | PU_CLERK | INDEFINIDO |
| Curtis | ST_CLERK | INDEFINIDO |
| Lex | AD_VP | INDEFINIDO |
| Julia | SH_CLERK | INDEFINIDO |
| Jennifer | SH_CLERK | INDEFINIDO |
| Louise | SA_REP | INDEFINIDO |
| Bruce | IT_PROG | PROGRAMADOR |
| Alberto | SA_MAN | INDEFINIDO |
| Britney | SH_CLERK | INDEFINIDO |
| Daniel | FI_ACCOUNT | CONTADOR |
| Pat | MK_REP | INDEFINIDO |
| Kevin | SH_CLERK | INDEFINIDO |
| Jean | SH_CLERK | INDEFINIDO |
| Tayler | SA_REP | INDEFINIDO |
| Adam | ST_MAN | INDEFINIDO |
| Timothy | SH_CLERK | INDEFINIDO |
| Ki | ST_CLERK | INDEFINIDO |
| Girard | SH_CLERK | INDEFINIDO |
| William | AC_ACCOUNT | INDEFINIDO |
| Douglas | SH_CLERK | INDEFINIDO |
| Kimberely | SA_REP | INDEFINIDO |
| Nancy | FI_MGR | INDEFINIDO |
| Danielle | SA_REP | INDEFINIDO |
| Peter | SA_REP | INDEFINIDO |
| Michael | MK_MAN | INDEFINIDO |
| Shelley | AC_MGR | INDEFINIDO |
| Guy | PU_CLERK | INDEFINIDO |
| Alexander | IT_PROG | PROGRAMADOR |
| Alyssa | SA_REP | INDEFINIDO |
| Charles | SA_REP | INDEFINIDO |
| Vance | SH_CLERK | INDEFINIDO |
| Payam | ST_MAN | INDEFINIDO |

**50 rows selected**

No exemplo abaixo, nós queremos retornar o salário incrementado de acordo com o tipo de cargo.

```
SELECT FIRST_NAME, JOB_ID, SALARY,
    DECODE(    JOB_ID,'IT_PROG',SALARY * 1.1,
    'FI_ACCOUNT',SALARY * 1.2,
    'AD_VP', SALARY * 0.95,
SALARY) DECODE_CARGO
FROM EMPLOYEES;
```

| FIRST_NAME | JOB_ID | SALARY | DECODE_CARGO |
|---|---|---|---|
| Steven | AD_PRES | 24000 | 24000 |
| Neena | AD_VP | 17000 | 16150 |
| Lex | AD_VP | 17000 | 16150 |
| Alexander | IT_PROG | 9000 | 9900 |
| Bruce | IT_PROG | 6000 | 6600 |
| David | IT_PROG | 4800 | 5280 |
| Valli | IT_PROG | 4800 | 5280 |
| Diana | IT_PROG | 4200 | 4620 |
| Nancy | FI_MGR | 12008 | 12008 |
| Daniel | FI_ACCOUNT | 9000 | 10800 |
| John | FI_ACCOUNT | 8200 | 9840 |
| Ismael | FI_ACCOUNT | 7700 | 9240 |
| Jose Manuel | FI_ACCOUNT | 7800 | 9360 |
| Luis | FI_ACCOUNT | 6900 | 8280 |
| Den | PU_MAN | 11000 | 11000 |
| Alexander | PU_CLERK | 3100 | 3100 |
| Shelli | PU_CLERK | 2900 | 2900 |
| Sigal | PU_CLERK | 2800 | 2800 |
| Guy | PU_CLERK | 2600 | 2600 |
| Karen | PU_CLERK | 2500 | 2500 |
| Matthew | ST_MAN | 8000 | 8000 |
| Adam | ST_MAN | 8200 | 8200 |
| Payam | ST_MAN | 7900 | 7900 |
| Shanta | ST_MAN | 6500 | 6500 |
| Kevin | ST_MAN | 5800 | 5800 |
| Julia | ST_CLERK | 3200 | 3200 |
| Irene | ST_CLERK | 2700 | 2700 |
| James | ST_CLERK | 2400 | 2400 |
| Steven | ST_CLERK | 2200 | 2200 |
| Laura | ST_CLERK | 3300 | 3300 |
| Mozhe | ST_CLERK | 2800 | 2800 |
| James | ST_CLERK | 2500 | 2500 |
| TJ | ST_CLERK | 2100 | 2100 |
| Jason | ST_CLERK | 3300 | 3300 |
| Michael | ST_CLERK | 2900 | 2900 |
| Ki | ST_CLERK | 2400 | 2400 |
| Hazel | ST_CLERK | 2200 | 2200 |
| Renske | ST_CLERK | 3600 | 3600 |
| Stephen | ST_CLERK | 3200 | 3200 |
| John | ST_CLERK | 2700 | 2700 |
| Joshua | ST_CLERK | 2500 | 2500 |
| Trenna | ST_CLERK | 3500 | 3500 |
| Curtis | ST_CLERK | 3100 | 3100 |
| Randall | ST_CLERK | 2600 | 2600 |
| Peter | ST_CLERK | 2500 | 2500 |
| John | SA_MAN | 14000 | 14000 |
| Karen | SA_MAN | 13500 | 13500 |
| Alberto | SA_MAN | 12000 | 12000 |
| Gerald | SA_MAN | 11000 | 11000 |
| Eleni | SA_MAN | 10500 | 10500 |

**50 rows selected**

A função condicional case permite o processamento condicional que exija o tratamento de várias hipóteses.

O exemplo seguinte avalia duas expressões lógicas e ainda oferece uma terceira possibilidade, quando as duas anteriores resultarem falsas:

```
SELECT FIRST_NAME, JOB_ID, SALARY,
   CASE
        WHEN SALARY < 5000 THEN 'AUMENTO'
        WHEN SALARY > 10000 THEN 'VERIFICAR'
ELSE 'NÃO AUMENTAR'
   END CLASSIFICAÇÃO
FROM EMPLOYEES;
```

| FIRST_NAME | JOB_ID | SALARY | CLASSIFICAÇÃO |
|---|---|---|---|
| Steven | AD_PRES | 24000 | VERIFICAR |
| Neena | AD_VP | 17000 | VERIFICAR |
| Lex | AD_VP | 17000 | VERIFICAR |
| Alexander | IT_PROG | 9000 | NÃO AUMENTAR |
| Bruce | IT_PROG | 6000 | NÃO AUMENTAR |
| David | IT_PROG | 4800 | AUMENTO |
| Valli | IT_PROG | 4800 | AUMENTO |
| Diana | IT_PROG | 4200 | AUMENTO |
| Nancy | FI_MGR | 12008 | VERIFICAR |
| Daniel | FI_ACCOUNT | 9000 | NÃO AUMENTAR |
| John | FI_ACCOUNT | 8200 | NÃO AUMENTAR |
| Ismael | FI_ACCOUNT | 7700 | NÃO AUMENTAR |
| Jose Manuel | FI_ACCOUNT | 7800 | NÃO AUMENTAR |
| Luis | FI_ACCOUNT | 6900 | NÃO AUMENTAR |
| Den | PU_MAN | 11000 | VERIFICAR |
| Alexander | PU_CLERK | 3100 | AUMENTO |
| Shelli | PU_CLERK | 2900 | AUMENTO |
| Sigal | PU_CLERK | 2800 | AUMENTO |
| Guy | PU_CLERK | 2600 | AUMENTO |
| Karen | PU_CLERK | 2500 | AUMENTO |
| Matthew | ST_MAN | 8000 | NÃO AUMENTAR |
| Adam | ST_MAN | 8200 | NÃO AUMENTAR |
| Payam | ST_MAN | 7900 | NÃO AUMENTAR |
| Shanta | ST_MAN | 6500 | NÃO AUMENTAR |
| Kevin | ST_MAN | 5800 | NÃO AUMENTAR |
| Julia | ST_CLERK | 3200 | AUMENTO |
| Irene | ST_CLERK | 2700 | AUMENTO |
| James | ST_CLERK | 2400 | AUMENTO |
| Steven | ST_CLERK | 2200 | AUMENTO |
| Laura | ST_CLERK | 3300 | AUMENTO |
| Mozhe | ST_CLERK | 2800 | AUMENTO |
| James | ST_CLERK | 2500 | AUMENTO |
| TJ | ST_CLERK | 2100 | AUMENTO |
| Jason | ST_CLERK | 3300 | AUMENTO |
| Michael | ST_CLERK | 2900 | AUMENTO |
| Ki | ST_CLERK | 2400 | AUMENTO |
| Hazel | ST_CLERK | 2200 | AUMENTO |
| Renske | ST_CLERK | 3600 | AUMENTO |
| Stephen | ST_CLERK | 3200 | AUMENTO |
| John | ST_CLERK | 2700 | AUMENTO |
| Joshua | ST_CLERK | 2500 | AUMENTO |
| Trenna | ST_CLERK | 3500 | AUMENTO |
| Curtis | ST_CLERK | 3100 | AUMENTO |
| Randall | ST_CLERK | 2600 | AUMENTO |
| Peter | ST_CLERK | 2500 | AUMENTO |
| John | SA_MAN | 14000 | VERIFICAR |
| Karen | SA_MAN | 13500 | VERIFICAR |
| Alberto | SA_MAN | 12000 | VERIFICAR |
| Gerald | SA_MAN | 11000 | VERIFICAR |
| Eleni | SA_MAN | 10500 | VERIFICAR |

**50 rows selected**

# Atividades:

1. Pesquise na tabela locations (retorne CITY e STATE_PROVINCE), aplicando uma função de decisão no campo State_Province, seguindo a instrução abaixo.

| Se STATE_PROVINCE for igual a | Valor Retornado (Criar uma alias para a coluna de retorno Retorno_Decode) |
|---|---|
| Washington | A String 'Bem vindo a Casa Branca' |
| Texas | A String 'Olá Cowboy ' |
| California | O campo cidade (CITY) |
| New Jersey | O campo Street_Address |

2. Pesquise na tabela employees (retorne o last_name e o salary), usando uma função de decisão no campo Salary, seguindo a instrução abaixo.

| Salário | Novo Salário |
|---|---|
| >= 4200 e <=6000 | Aumento de 10% |
| >=9000 e <=17000 | Aumento de 20% |
| Outros casos | Retorna o salario atual |

# Material Complementar

Para ajudar a criar toda a estrutura de relacionamento entre as tabelas e gerar os scripts necessários existem esses dois documentos oficiais disponíveis.

- **Erwin** – http://support.ca.com/cadocs/0/e002953e.pdf.
- **Dbdesigner** – http://downloads.mysql.com/DBDesigner4/DBDesigner4_manual_1.0.42.pdf.

# Referências

FANDERUFF, Damaris. **Dominando o Oracle 9i: Modelagem e desenvolvimento.** São Paulo: Pearson Education do Brasil, 2003.

MORELLI, Eduardo M. Terra, 1996. **Oracle 9i Fundamental: Sql, Pl/SQL e Administração.** São Paulo: Érica, 2002.

Cruzeiro do Sul
Educacional