03

# Filtrando e Ordenando dados

## Filtrando e ordenando dados

Vamos falar agora sobre a ordem dos resultados do SELECT. Vamos executar a seguinte *query*:

```
SQL> select nome from funcionarios;

NOME
--------------------------------------------------------------
Kaun
Jose
Eduarda
Leonardo
Thais
Nicole
Leticia
Joao
Igor
Diogo
Karlos

NOME
--------------------------------------------------------------
Stevan
Otavio

13 linhas selecionadas.
```

Qual é a ordem em que os nomes aparecem? É de inserção ou de a partir do maior salário? Observe que quando executamos o `select` e não definimos a ordem, os dados poderão ser listados com qualquer ordenação. Eles não terão uma ordem específica. Ela será definida pelo algoritmo do Oracle, que irá buscar os dados o mais rápido possível.

Então, se não definimos uma ordem, o Oracle a define.

Para alterarmos a ordem iremos usar o `order by` . Se quisermos ordenar os funcionários pela ordem de salário, selecionaremos as colunas `nome` e `salario` .

```
SQL> select nome,salario from funcionarios order by salario;
```

Se executarmos, ele irá mostrar os salários ordenados do mais baixo para o mais alto.

Porém, se estou montando um ranking, quero que o maior salário seja o primeiro a ser listado. Não foi isto o que aconteceu, por quê? Porque quando usamos o `order by` ficar um termo implícito na linha: o `asc` . O termo significa que a ordem será ascendente. Então, se adicionarmos `desc` significa que estamos pedindo para a ordem ser decrescente.

```
SQL> select nome,salario from funcionarios order by salario desc;
```

Agora, o primeiro nome listado será o do Igor, que tem um salário de R$6700, e o último o Kauan, que recebe R$1100.

É possível perceber que temos muitos salários repetidos. Podemos adicionar que os salários sejam distintos.

```
SQL> select distinct salario, nome from funcionarios order by salario desc;

   SALARIO
----------
NOME
--------------------------------------------------------
      6700
Igor
      6700
Thais
      3700
Leonardo
```

No entanto, podemos perceber que ele não ocultou as repetições. Por que isto aconteceu? Porque o `distinct` é aplicado à linha e o `order by` irá afetar isto.

Vamos testar o `distinct` calculando a soma `salario + vale_refeicao`.

```
SQL> select distinct salario + vale_refeicao, nome from funcionarios order by salario desc;
select distinct salario + vale_refeicao, nome from funcionarios order by salario desc

ERRO na linha 1:
ORA-01791: nao ha uma expressao de SELECT
```

Toda vez que usamos o `distinct` em um campo no qual estamos calculando, teremos que usar o campo no `order by`. Vamos adicionar `salario + vale_refeicao` no `order by`.

```
SQL> select distinct salario + vale_refeicao, nome from funcionarios order by salario + vale_refeica

SALARIO+VALE_REFEICAO
---------------------
NOME
--------------------------------------------------------------------------------
karlos

Otavio

Stevan

SALARIO+VALE_REFEICAO
---------------------
NOME
--------------------------------------------------------------------------------
                 6700
Igor
                 6700
Thais

                 4000
```

```
Leonardo
```

Esta é a parte inicial da lista. Porém, `salario + vale_refeicao` chamamos de `custo`. Vamos alterar o nome e depois, incluí-lo no `order by`.

```
SQL> select distinct salario + vale_refeicao as custo, nome from funcionarios order by custo desc;
```

E tudo irá funcionar perfeitamente.

```
    CUSTO
--------
NOME
------------------------------------------------------------------
     6700
Igor

     6700
Thais

     4000
Leonardo
```

Como a ordem é elaborado após os resultados serem trazidos do banco de dados, o **alias** já terá sido aplicado. Então, por isso, conseguimos usar o alias dentro do `order by` - o mesmo não irá acontecer dentro do `where`.

Vamos agora pedir para ele mostrar os valores do vale-refeição. Pediremos também que a ordenação seja feita a partir dos maiores valores da coluna `vale_refeicao`.

```
SQL> select salario + vale_refeicao as custo + vale_refeicao from funcionarios order by vale_refeica

    CUSTO VALE_REFEICAO
-------- -------------



     4000           300
     2000           300
     3700           300
     1800           300
     1500           300
     1500           300
     1400           300
     2000           300

    CUSTO VALE_REFEICAO
-------- -------------
     6700             0
     6700             0

13 linhas selecionadas.
```

Obserse que as primeiras linhas vazias. Todos os funcionário que estão nulos, foram jogados para o topo da tabela. Isto acontece, porque temos as palavras `null first` implícitas na *query*, por padrão.

```
SQL> select salario + vale_refeicao as custo + vale_refeicao from funcionarios order by vale_refeica
```

Mas se queremos que estes funcionários passem a ser exibidos no fim da tabela, vamos inverter a ordem usando o `null last`.

```
SQL> select salario + vale_refeicao as custo + vale_refeicao from funcionarios order by vale_refeica

   CUSTO VALE_REFEICAO
-------- -------------
    4000           300
    2000           300
    3700           300
    1800           300
    1500           300
    1500           300
    1400           300
    2000           300
    6700             0
    6700             0

   CUSTO VALE_REFEICAO
-------- -------------


13 linhas selecionadas.
```

Quem está nulo ficou nas linhas vazias, embaixo.

Então, nós já aprendemos a definir uma ordem e que podemos usar o alias dentro da ordem, além de que, quando temos o `distinct` em um campo precisamos adicionar o mesmo campo no `order by`. Ainda aprendemos como fazer com que os valores nulos sejam exibidos no fim do resultado.

Em seguida, veremos como limitar os nossos resultados.

## Limitando os nossos resultados

Vamos definir a ordem do ranking.

```
SQL> select salario from funcionarios order by salario desc;

   SALARIO
----------
      6700
      6700
      3700
      3400
```

```
      1700
      1700
      1500
      1500
      1500
      1500
      1200

   SALARIO
----------
      1200
      1100

13 linhas selecionadas.
```

O SQL *Plus irá trazer 13 registros e irá ordenar os salários do maior para o menor. Eu quero dar uma bonificação para os cinco funcionários que ganham mais na empresa. Para fazer isto, preciso limitar o ranking. Farei isto adicionando ao SELECT:

```
fetch first 5 rows only
```

Apenas os cinco maiores salários serão listados.

```
SQL> select salario from funcionarios order by salario desc fetch first 5 rows only;

   SALARIO
----------
      6700
      6700
      3700
      3400
      1700
```

Os dois funcionários que recebem R$6700 provavelmente são líderes de equipe. Eu gostaria de descartá-los. Para isto, antes de usarmos o `fetch`, adicionaremos `offset 2 rows`, porque queremos "picotar" as duas primeiras linhas. Iremos também alterar outro ponto da *query*. Em vez de listarmos as 5 primeiras ( `first` ) linhas, iremos listar as próximas ( `next` ) 5 linhas.

```
SQL> select salario from funcionarios order by salario desc offset 2 rows fetch next 5 rows only;

   SALARIO
----------
      3700
      3400
      1700
      1700
      1500
```

Agora, ele descartou os dois registros `6700` e exibiu os cinco seguintes.

No entanto, será que o só uma pessoa que recebe o salário de R$1500. Pela nossa lista, não temos mais como saber. Mas se torna injusto, se outras pessoas receberem o mesmo valor de salário e não ganharem a bonificação. O correto seria pedir

para "considerar os empates". Quando pedimos para ele selecionar apenas as 5 linhas, pedimos para que os empates sejam ignorados. Então, iremos pedir que ele considere as 5 linhas com empates ( `with ties` ).

```
SQL> select salario from funcionarios order by salario desc offset 2 rows fetch next 5 rows with tie

   SALARIO
----------
      3700
      3400
      1700
      1700
      1500
      1500
      1500
      1500

8 linhas selecionadas.
```

Ao executarmos, ele irá trazer 8 linhas, porque ele exibiu todos os registros que empatam com o último resultado.

Existe outra forma de selecionarmos os registros. Podemos pesquisar a porcentagem dos funcionários. Vamos testar com os 5 porcento ( `percent` ) que ganham os melhores salários, ainda desconsiderando as duas primeiras linhas.

```
SQL> select salario from funcionarios order by salario desc offset 2 rows fetch next 5 percent rows

   SALARIO
----------
      3700
```

Se no total, temos 13 registros, vamos desconsiderar os dois primeiros. Os 5% serão calculados de 11 funcionários. Se pesquisássemos os 25%, ele retornaria quatro registros.

```
SQL> select salario from funcionarios order by salario desc offset 2 rows fetch next 5 percent rows

   SALARIO
----------
      3700
      3400
      1700
      1700
```

Além de especificar por linhas, podemos especificar a busca usando porcentagens. Observe que no cálculo da porcentagem, ele sempre arredondará para cima. Por exemplo, se 5% for igual a 0,5, o SQL irá retornar um registro.

Nós já aprendemos a cortar os resultados, usar o `offset` e colocar um limite. Incluir resultados com e sem empate, considerar por linha ou limitar por porcentagem. Nós já sabemos usar o `where` , `fetch` , e `offset` e outras funções, o que nos permite fazer mais consultas poderosas.

Além de utilizarmos o SELECT para mostrar os dados, conseguimos gerar alguns tipos de relatórios baseado no conhecimentos que estamos adquirindo.

Até o próximo curso de certificação!