04

Entendendo funções de agrupamento

Entendendo funções de agrupamento

No último vídeo, estudamos como funciona o `GROUP BY`, agora estudaremos outras funções de agrupamento.

Ainda iremos trabalhar com a tabela `funcionarios`. Quando trabalhamos com valores de salários, talvez a métrica que utilizamos mais cotidianamente seja a média salarial de uma empresa. No Oracle, temos uma função específica para fazermos este cálculo: é a `avg()`.

`AVG` é uma abreviação do termo em inglês *average*, que traduzido para Português significa *média*. Para conhecermos seu compoartamento, usaremos a função com os valores da coluna `salario`.

```
SQL> select avg(salario) from funcionarios;

AVG(SALARIO)
------------
  2939,28571
```

A média de salário da empresa é de `R$2.939,28`.

Como não definimos um `group by()`, ele usou todos os dados como um único grupo. E se quisermos saber a média por setor da minha empresa? Para isto, precisaremos definir o agrupamento com `group by`.

```
SQL> select avg(salario),setor_id from funcionarios group by setor_id;

AVG(SALARIO)   SETOR_ID
------------ ----------
        3500          1
        2700          2
        2537,5        3
```

Ele retornou a média salarial dos setores `1,2 e 3`. Observe que a função de agrupamento tem um comportamento semelhante ao da `count()`, ainda que tenha um uso específico.

Vamos pensar em **detalhes**, porque eles serão cobrados no exame da certificação. Iremos pensar na hipótese de calcularmos a média dos nomes. Não faria muito sentido, porque se trata de texto... Se tentarmos encontrar a média de `nome`, o Oracle irá retornar uma mensagem de erro.

```
SQL> select avg(nome) from funcionarios;
select avg(nome) from funcionarios

ERRO na linha 1:
ORA-01722: numero invalido
```

Logo, podemos concluir que a função `avg()` irá pedir um coluna **numérica**.

E quando calculamos a média. será que ele considerou repetições? Por padrão, sim. O Oracle irá considerar todos os registros, incluindo as repetições.

Dentro da minha função, posso adicionar uma cláusula informando se quero **incluir** ou **excluir** as repetições. Por padrão, ele irá incluir e dentro do parênteses teremos oculto o termo `all`.

```
SQL> select avg(all salario) from funcionarios;

AVG(ALLSALARIO)
---------------
     2939,28571
```

O resultado foi o mesmo que a média anterior, quando ele também incluiu as repetições. Para que ele desconsidere os valores repetidos, teremos que adicionar `distinct` na função `avg()`.

```
SQL> select avg(distinct salario) from funcionarios;

AVG(DISTINCTSALARIO)
--------------------
              3062,5
```

A média agora ficou superior ao resultado anterior, o que faz sentindo, considerando que os valores que mais repetem são os menores salários.

Porém, às vezes descobrir a média de alguns cálculos mais complicados. Por exemplo, na nossa tabela, temos além dos valores do `salario`, a `pct_comissao` (porcentagem da comissão). Então, o valor real de um `salario`, inclui a porcentagem do salário? Mas como fazemos este cálculo. Faremos um `select` de `salario` somado a `salario * pct_comissao`. Como queremos a média, colocaremos a conta dentro do `avg()`.

```
SQL> select avg(salario + salario * pct_comissao) from funcionarios;

AVG(SALARIO+SALARIO*PCT_COMISSAO)
---------------------------------
                             2745
```

Ele irá retornar o Valor `2745`. Vamos verificar se o resultado está correto. Para isto, iremos selecionar apenas a conta `salario + salario * pct_comissao`.

```
SQL> select salario + salario * pct_comissao from funcionarios;

SALARIO+SALARIO*PCT_COMISSAO
----------------------------
                        3300
                        2420

                        1980
                        2860


                        2200
```

```
                         1600
                         2640
                         2310

SALARIO+SALARIO*PCT_COMISSAO
----------------------------
                         2420
                         3795
                         4620

14 linhas selecionadas.
```

Veremos que vários registros retornaram com um valor nulo. Isto acontece, por alguns funcionários recebiam apenas salário e não recebiam comissão. Nestes casos, o valor de `pct_comissao` é `null`, o que faz com que o resultado da conta seja nulo também.

Qual é o comportamento da função `AVG` quando encontra uma linha nula? A função irá ignorá-la e esta não será incluída na média. Ignorar a linha nula é o comportamento padrão da maioria das funções de agrupamento.

Mas vimos anteriormente que temos funções para trabalharmos com valores nulos. Nós iremos usar a `nvl()` e quando os valores da comissão forem nulos, quero que ele se transforme em `0` (zero). Quando executarmos a *query*, a tabela estará toda preenchida.

```
SQL> select salario + salario * nvl(pct_comissao,0) from funcionarios;

SALARIO+SALARIO*NVL(PCT_COMISSAO,0)
-----------------------------------
                               3300
                               2420
                               2200
                               1980
                               2860
                               6200
                               5300
                               2200
                               1650
                               2640
                               2310

SALARIO+SALARIO*NVL(PCT_COMISSAO,0)
-----------------------------------
                               2420
                               3795
                               2620

14 linhas selecionadas.
```

Agora, nós conseguiremos calcular a média disto, adicionando a função `avg()` à *query*.

```
SQL> select avg(salario + salario * nvl(pct_comissao,0)) from funcionarios;

AVG(SALARIO+SALARIO*NVL(PCT_COMISSAO,0)
```

```
-------------------------------------
                            3135,35714
```

Nós conseguimos trabalhar com as funções de agrupamento juntamente com as de linha única.

Havia explicado a possibilidade de ter questões "pegadinhas" sobre nulo. Este é um caso interessante e pode ser cobrado no exame de certificação.

Além de calcular a média, nós já havíamos conhecido outra função de agrupamento `count()`. O uso mais comum desta função é `count(*)`.

```
SQL> select count(*) from funcionarios;

  COUNT(*)
----------
        14
```

A função `count()` conta a quantidade de registros. O `*` (asterisco) indica que queremos contar a quantidade de registros independentemente se temos colunas nulas ou não. Anteriormente, citamos que a maioria das funções ignoram valores nulos. O `COUNT` tem este mesmo comportamento. Quando usamos o `*`, ele irá retornar a quantidade de registros encontrados incluindo a coluna nula.

E quando é interessante usar o `count()`? Podemos, por exemplo, contar quantas pessoas recebem determinado salário. Para descobrirmos a resposta, usaremos `count(salario)`.

```
SQL> select count(salario) from funcionarios;

COUNT(SALARIO)
--------------
            14
```

Como não temos nenhum valor nulo na coluna `salario`, ele retornou o número total de salários. Se testássemos com um coluna que possui valores nulos, ele iria ignorar os resultados nulos.

Vale ressaltar, que quando usamos o `COUNT`, o comportamento padrão é que a função considere todos os registros, considerando inclusive os valores que se repetem. Mas assim com a função `AVG`, o `COUNT` também tem um `all` omitido no parênteses.

```
SQL> select count(all salario) from funcionarios;

COUNT(ALLSALARIO)
-----------------
               14
```

Mas se incluirmos um `distinct` na *query*, ele irá retornar apenas a quantidade de registros distintos.

```
SQL> select count(distinct salario) from funcionarios;

COUNT(DISTINCTSALARIO)
```

```
----------------------
                    12
```

Ele retornou `12`, porque dois salários se repetem na empresa.

Outro uso comum é `count(id)`.

```
SQL> select count(id) from funcionarios;

 COUNT(ID)
----------
        14

SQL> desc funcionarios;
 Nome                                      Nulo?    Tipo
 ----------------------------------------- -------- ----------------
 ID                                        NOT NULL NUMBER(11)
 NOME                                               VARCHAR2(20)
 SALARIO                                            NUMBER(10,2)
 PCT_COMISSAO                                       NUMBER(10,2)
 SETOR_ID                                           NUMBER(11)


O `id` não é um valor nulo e será do tipo `number()`. O `id` é a nossa **chave primária** - que tem


Podemos também usá-la juntamente com o `group by()`:
```

SQL> select count(id) from funcionarios group by setor_id;

COUNT(ID)

5 5 4

```
Ele retornou a quantidade de funcionários para cada setor.

Nós conhecemos mais duas função de agrupamento: `COUNT` e `AVG`. Iremos conhecer outras adiante.

###Outras funções de agrupamento

Continuando com as funções de agrupamento, podemos querer extrair mais dados da tabela `funcionarios

Agora, queremos descobrir qual é o **maior** e o **menor** salário. Nós poderíamos visualizar a tabe

Se queremos encontrar o maior salário, podemos usar a função `max()`:
```

SQL> select max(salario) from funcionarios;

MAX(SALARIO)

6200

O maior salário é `6200`. Para encontrarmos o menor salário, temos a função `min()`. Antes de execut

SQL> select max(salario) as MaiorSalario, min(salario) as MenorSalario from funcionarios;

MAIORSALARIO	MENORSALARIO

6200 1500

Conseguimos extrair também que o menor salário é `1500`. Vamos começar a pensar como estas funções s

SQL> select max(null) as MaiorSalario, min(salario) as MenorSalario from funcionarios;

M MENORSALARIO

1500

Ele irá retornar valores nulos. O mesmo irá acontecer se usamos uma lista que só contenha valores nu

Observe que as duas funções, `max()` e `min()` irá retornar um único valor. E se adicionarmos o `gro

SQL> select max(salario) as MaiorSalario, min(salario) as MenorSalario from funcionarios group by setor_id;

MAIORSALARIO	MENORSALARIO

6200 1500 5300 1800 3450 2100

Ele ordenou os resultados a partir do primeiro setor.

E o que acontece se passarmos para a função `max()` um campo de texto? Faremos o teste com a coluna

SQL> select max(nome) from funcionarios;

MAX(NOME)

Tereza

Por que ele retornou `Tereza`? Tanto a função `MIN` como a `MAX`, quando executadas em um campo de t

SQL> select min(nome) from funcionarios;

```
MIN(NOME)
--------------------
Alexandre
```

Então, podemos aplicar a função em campos numéricos, de texto e com datas.

Podemos ainda querer gerar um relatório de custos para minha empresa, por exemplo, descobrir o gasto total. Para isto, iremos somar os gastos com a função `sum()`. Em seguida, vamos usar a função `sum()` com `salario`.

```
SQL> select sum(salario) from funcionarios;

SUM(SALARIO)
------------
       41150
```

Apenas com salários, a empresa gasta R$41.150. Vamos agora incluir no cálculo a porcentagem de comissão.

```
SQL> select sum(salario + salario * pct_comissao) from funcionarios;

SUM(SALARIO+SALARIO*PCT_COMISSAO)
---------------------------------
                            30195
```

Apesar de incluir a comissão, o valor total ficou inferior. Aconteceu algo semelhante ao que vimos com a função `avg()` (de média). Quando a função `sum()` encontra algum valor nulo na listagem, ele simplesmente ignora. Então, novamente, iremos incluir a função `nvl()` para podermos transformar valores nulos em `0`.

```
SQL> select sum(salario + salario * nvl(pct_comissao,0)) from funcionarios;

SUM(SALARIO+SALARIO*NVL(PCT_COMISSAO,0))
----------------------------------------
                                   43895
```

Podemos também levantar os gastos com cada setor. Para isto, iremos definir a regra de agrupamento com `group by`. Teremos o subtotal para cada setor.

```
SQL> select sum(salario + salario * nvl(pct_comissao,0)) from funcionarios group by setor_id;

SUM(SALARIO+SALARIO*NVL(PCT_COMISSAO,0))
----------------------------------------
                                   18630
                                   14320
                                   10945
```

E para excluirmos as repetições dos gastos? Neste caso, o `sum()` terá um comportamento análogo ao que vimos nas funções de média e `count()`. Dentro do parênteses, temos o `all` omitido, mas se incluímos o `distinct`, ele irá retirar os valores repetidos.

```
SQL> select sum(distinct salario) from funcionarios;

SUM(DISTINCTSALARIO)
--------------------
               36750
```

Com alguns registros a menos, o resultado foi inferior ao anterior.

Anteriormente, vimos que função `avg()` calcula a média de salário.

```
SQL> select avg(salario) from funcionarios;

AVG(SALARIO)
------------
  2939,28571
```

Temos está média, mas que ficou muito distante do maior valor registrado na tabela. Valores muito altos ou pequenos, acabam criando estas distinções e a média pode não representar de forma real os dados. Por isso, quando trabalhamos com estatística, trabalhamos com a **mediana**.

Iremos visualizar um exemplo.

```
SQL> 20 30 45 80 1020 5000 5200
```

Se tirarmos a médias apenas dos três valores finais (`1020, 5000, 5200`), será um valor muito diferente da média dos três primeiros valores (`20,30,45`). O que fazemos com a mediana é selecionar o valor que está no meio de todos, no nosso exemplo, é `80` .

Para calcularmos a mediana, usaremos a função `median()` de `salario` .

```
SQL> select median(salario) from funcionarios;

MEDIAN(SALARIO)
---------------
           2300
```

O valor da mediana é `2300` , um pouco abaixo do resultado anterior. E o que acontece se tentamos descobrir a mediana de um campo de texto? Ele irá retornar que o argumento deve ser do tipo numérico, data ou horário.

```
SQL> select median(nome) from funcionarios;
select median(nome) from funcionarios

ERRO na linha 1:
ORA-30495: O argumento deve ser do tipo numerico ou data/data/horario.
```

A função `median()` não tem a propriedade de trabalhar com texto, como vimos com a `max()` e `min()` . Ela fnciona apenas com **campos numéricos**.

Neste vídeo conhecemos mais algumas funções de agrupamento. Em seguida, conheceremos outras.

Continuando com funções de agrupamento

Veremos duas funções que são importantes quando trabalhamos com estatísticas. A primeira delas irá calcular o quanto os nossos dados variam em relação à média.

Iremos visualizar a listagem dos salários.

```
SQL> select salario from funcionarios;

   SALARIO
----------
      3000
      2200
      2200
      1800
      2600
      6200
      5300
      2000
      1500
      2400
      2100

   SALARIO
----------
      2200
      3450
      4200

14 linas selecionadas.
```

Vamos calcular a média com a função `avg()` .

```
SQL> select avg(salario) from funcionario;

AVG(SALARIO)
------------
  2939,28571
```

Vimos os salários listados e a média, agora queremos calcular o **desvio padrão**. Esta é um métrica usada em estatística que consegue identificar o quanto os salários variam em relação à média. No Oracle, temos uma função que calcula o desvio padrão: `stddev()` , que é a abreviação de *standard deviation*.

```
SQL> select stddev(salario) from funcionarios;

STDDEV(SALARIO)
---------------
     1388,43833
```

O Oracle retornou que o desvio padrão é `1388,43833` . Vamos arredondar para duas casa antes da vírgula, usando a função `round()` .

```
SQL> select round(stddev(salario),-2) from funcionarios;

ROUND(STDDEV(SALARIO),-2)
-------------------------
                     1400
```

O valor que o Oracle retornou indica que os salários oscilam R$1400 para mais ou para menos. Vamos testar se o resultado está certo.

Vamos calcular a média com `avg(salario)` somado ao desvio padrão e iremos chamá-lo como `Maximo` .

```
SQL> select (avg(salario) + round(stddev(salario),-2)) as Maximo from funcionarios;
```

Queremos descobrir o valor mínimo também. Iremos calcular a média e subtraí-la pelo desvio padrão, que chamaremos de `Minimo` .

```
SQL> select(avg(salario) + round(stddev(salario,-2)) as Maximo, (avg(salario) - round(stddev(salaric

    MAXIMO     MINIMO
---------- ----------
4330,28571 1539,28571
```

Agora, os valores ficaram mais próximos. Se observarmos a listagem mais acima de salários, veremos que o maior salário (R$6200) foge da mediana, mas em relação aos demais valores, os resultados de máximo e mínimo estão próximos.

Existe outra medida, bastante relacionada ao desvio padrão que é a **variancia**, que podemos calcular com a função `variance()` .

```
SQL> select variance(salario) from funcionarios;

VARIANCE(SALARIO)
-----------------
       1927760,99
```

O devio padrão é o resultado da raiz quadrada da variância. Podemos verificar isso, usando a função `strt()` (*square root*) e veremos o resultado:

SQL> select sqrt(variance(salario)) as DevioPadrao from funcionarios;

DESVIOPADRAO

1388,43833

``` O Oracle retornou exatamente o mesmo valor do desvio padrão. Então, esta é a relação entre variância e desvio padrão. As duas funções são bastante relevantes quando trabalhamos com estatísticas, mas é improvável que caiam no exame de
```

certificação.

Na próxima aula, veremos como filtrar os dados quando trabalhamos co funções de agrupamento.