



Funções de Grupo



**Curso de Introdução a Oracle 11g:
SQL/Avançado**

Prof.: Marlon Mendes Minussi
marlonminussi@gmail.br



O que são funções de Grupo

- Diferente das funções básicas (single-row), as funções de grupo atuam em conjuntos de linhas para obter um resultado por grupo.
- Estes conjuntos podem ser tabelas inteiras ou a tabela dividida em grupos.



Tipos de Funções de Grupo

- Cada uma das funções aceita um argumento.
- A tabela abaixo identifica as opções que podem ser utilizadas na sintaxe.

Função

AVG([DISTINCT|ALL] n)

COUNT({[DISTINCT|ALL]expr})

MAX([DISTINCT|ALL] expr)

MIN([DISTINCT|ALL] expr)

STDDEV([DISTINCT|ALL] n)

SUM([DISTINCT|ALL] n)

VARIANCE ([DISTINCT|ALL]n)

Descrição

Valor médio n, ignorando valores nulos

Nº de linhas, onde expr computa somente valores diferentes de nulo. Para contar todas as linhas selecionadas utilize *, incluindo linhas duplicadas e com valores nulos.

Valor máximo de expr, ignorando valores nulos.

Valor mínimos de expr, ignorando valores nulos..

Desvio padrão de n, ignorando valores nulos.

Soma dos valores de n, ignorando valores nulos.

Variância dos valores de n, ignorando valores nulos.



Utilizando Funções de Grupo

- Sintaxe:
SELECT group_funciton(column)
FROM table
[WHERE condition]
[ORDER BY
 expr];
- Diretrizes para utilização da Função de Grupo.
 - DISTINCT faz a função validar somente valores não duplicados; ALL faz a função considerar todos os valores, inclusive os duplicados. O padrão é ALL e portanto não precisa ser especificado.
 - Os tipos de dados para os argumento podem ser CHAR, VARCHAR2, NUMBER OU DATE, onde expr for especificado.
 - Todas as funções de grupo exceto COUNT(*) ignoram valores nulos. Para substituir um valor nulo, utilize a função NVL.



AVG e SUM

- Ex.:

```
SELECT AVG(valor), MAX(valor), MIN(valor), SUM(valor)  
FROM atendimento;
```
- Você pode utilizar as funções AVG, SUM, MIN e MAX em colunas que podem armazenar dados numéricos.
- O exemplo acima exibe média, o maior, o menor e a soma dos valores dos atendimento.



MIN e MAX

- Ex.:

```
SELECT MIN(dt_atendimento), MAX(dt_atendimento)  
FROM atendimento;
```
- Você pode utilizar as funções MAX e MIN para qualquer tipo de dado. O exemplo acima exibe o mais recente e o mais antigo atendimento.
 - Nota: As funções AVG, SUM, VARIANCE, STDDEV só podem ser utilizadas com tipos de dados numéricos.



COUNT

- Ex.:

```
SELECT COUNT(*)  
FROM atendimento;
```
- O exemplo acima exibe todos os atendimentos existentes.
- A função COUNT possui dois formatos:
 - COUNT(*) retorna o número de linhas em uma tabela, incluindo linhas duplicadas e linhas que contém valores nulos.
 - COUNT(expr), ao contrário, retorna o numero e linhas com valores diferentes de nulo na colunas identificada por expr.



count

- Ex.:

```
SELECT COUNT(desconto)  
FROM atendimento;
```
- O exemplo abaixo exibe o número de atendimentos que possuem desconto.
- Observe que o resultado fornece um número total menor que o número de atendimentos porque possui atendimentos sem descontos, logo existem valores nulos na coluna descontos.



MV [COUNT

- Ex.:

```
SELECT COUNT(DISTINCT(cod_pac))  
FROM atendimento;
```
- O exemplo acima mostra o número de pacientes atendidos na clínica.



NVL com Função de Grupo

- Ex.:

```
SELECT AVG(NVL(desconto,0))  
FROM atendimento;
```
- A função NVL força as funções de grupo a considerar os valores nulos no cálculo.
- No exemplo acima, a média é calculada baseada em todas as linhas da tabela embora existam valores nulos armazenados na coluna DESCONTO. A média é calculada dividindo o desconto total para todos os atendimentos pelo número total atendimentos.



GROUP BY

- Até agora, todas as funções de grupo trataram a tabela como um grande grupo de informação.
- Às vezes, você pode precisar dividir a tabela em grupos menores. Isto pode ser feito utilizando a cláusula GROUP BY.
- Sintaxe:
SELECT column, group_function (column)
FROM table
[WHERE condition]
[GROUP BY
 group_by_expression]
[ORDER BY
 expr];



GROUP BY

- Você pode utilizar a cláusula GROUP BY para dividir as linhas de uma tabela em grupos.
- Você pode então utilizar as funções de grupo para devolver informação sumarizada para cada grupo.
- Sintaxe:
group_by_expression especifica as colunas cujo valores determinam a base para agrupamento de linhas.

GROUP BY

- Diretrizes:

- Se você incluir uma função de grupo em um cláusula SELECT, você não pode selecionar resultados individuais a menos que a coluna individual apareça na cláusula GROUP BY. Você receberá uma mensagem de erro caso não inclua a coluna na lista.
- Utilizando a cláusula WHERE, você pode excluir linhas antes de fazer a divisão dos grupos,
- Você deve incluir as colunas na cláusula GROUP BY.
- Você não pode utilizar o alias de uma coluna na cláusula GROUP BY.
- Por default, as linhas são classificadas em ordem ascendente das colunas incluídas na lista GROUP BY. Você pode sobrepor esta ordenação utilizando a cláusula ORDER BY.

GROUP BY

- Ex.:

```
SELECT cod_pac, AVG(valor)  
FROM atendimento  
GROUP BY cod_pac  
ORDER BY 1;
```

```
SELECT AVG(valor)  
FROM atendimento  
GROUP BY cod_pac  
ORDER BY 1;
```

- Quando utilizar a cláusula GROUP BY, tenha certeza que todas as colunas de lista da cláusula SELECT que não estão em funções de grupo estejam na lista da cláusula GROUP BY.
- O exemplo acima exibe o número do cliente e a média do valor do atendimento de cada um.



GROUP BY em Múltiplas Colunas

- Ex.:

```
SELECT to_date(at.dt_atendimento,'DD-MM-YY') DATA, p.estado UF ,  
       SUM(at.valor)  
FROM atendimento at, paciente p  
WHERE at.cod_pac=p. cod_pac  
GROUP BY  
       to_date(at.dt_atendimento,'DD-MM-YY'),  
       p.estado  
ORDER BY 1
```

- Você pode retornar resultados sumarizados para grupos e subgrupos listando mais de uma coluna na cláusula GROUP BY. A ordem de classificação padrão dos resultados baseia-se na ordem das colunas da cláusula GROUP BY. A seguir é representado como o comando SELECT acima, contendo a cláusula GROUP BY, é avaliada:



GROUP BY em Múltiplas Colunas

- A cláusula SELECT especifica as colunas a serem recuperadas:
 - Data do Atendimento da tabela ATENDIMENTO
 - Estado na tabela PACIENTE
 - A soma de todos os valores na tabela ATENDIMENTO
 - A cláusula FROM especifica as tabelas que o banco de dados deve acessar: tabelas ATENDIMENTO E PACIENTE
 - A cláusula GROUP BY especifica como as linhas devem ser agrupadas:
 - Primeiro, as linhas são agrupadas através da data do atendimento.
 - Segundo, dentro dos grupos de datas de atendimento as linhas são agrupadas pelo estado do paciente.
 - A função SUM esta sendo aplicada à coluna valor para todos os atendimentos dentro de cada grupo de data e estado.



Consultas Ilegais

- Utilizando uma função de grupo dentro da cláusula WHERE.

- Ex.:

```
SELECT cod_pac, AVG(valor) MEDIA  
FROM atendimento  
WHERE AVG(valor) >= 100  
GROUP BY  
        cod_pac  
ORDER BY 1;
```



Consultas Ilegais

- A cláusula WHERE não pode ser utilizado para restringir grupos. O comando SELECT acima resulta em um erro porque utiliza a cláusula WHERE para restringir a exibição das médias do valor do atendimento por paciente que possuem média maior que 100.
- Ex.:

```
SELECT cod_pac, AVG(valor) MEDIA
FROM atendimento
GROUP BY
    cod_pac
HAVING AVG(valor) >= 100
ORDER BY 1;
```



HAVING

- Sintaxe:

```
SELECT column, group_function (column)
FROM table
[WHERE condition]
[GROUP BY
    group_by_expression]
[HAVING group_condition]
[ORDER BY
    expr];
```

- Você utiliza a cláusula HAVING para especificar quais grupos serão exibidos. Portanto, você restringe grupos baseado em informações agregadas.

Ex.:

```
SELECT dt_atendimento DATA , SUM(valor) TOTAL  
FROM atendimento  
WHERE (sysdate-dt_atendimento) < 7  
GROUP BY  
        dt_atendimento  
HAVING SUM(valor) >= 100  
ORDER BY 1;
```

- O exemplo acima exibe os atendimentos agrupados por data do atendimento, onde a diferença entre datas do atendimento e a data atual seja inferior a 7 dias (uma semana) e a soma destes atendimentos seja maior ou igual a 100 reais.



Exercícios GROUP BY e HAVING

1. Qual o total dos valores dos exames?

- `SELECT SUM(valor) FROM exame;`

2. Qual o nome do paciente o total de valores de atendimento?

- `SELECT p.nome, SUM(at.valor)
FROM paciente p, atendimento at
WHERE p.cod_pac=at.cod_pac
GROUP BY p.nome;`

3. Quantos atendimentos foram feitos?

- `SELECT COUNT(*) FROM atendimento;`

4. Quantos atendimentos cada medico efetuou?

- `SELECT m.nome, COUNT(*)
FROM medico m, atendimento at
WHERE m.cod_med=at.cod_med
GROUP BY m.nome;`



GROUP BY e HAVING

5. Qual o valor mínimo e o valor máximo dos valores dos exames?
 - `SELECT MIN(valor), MAX(valor) FROM exame;`
6. Qual o nome, valor mínimo e o valor máximo dos valores dos atendimentos de cada paciente?
 - `SELECT p.nome, MIN(at.valor), MAX(at.valor)
FROM paciente p , atendimento at
WHERE p.cod_pac=at.cod_pac
GROUP BY p.nome;`



GROUP BY e HAVING

7. Média do valor dos atendimentos feitos em dois determinados dias?

- ```
SELECT dt_atendimento, AVG(valor)
FROM atendimento
WHERE dt_atendimento IN ('30/03/11','31/03/11')
GROUP BY dt_atendimento;
```

8. Qual total de valores de atendimentos de cada medico que superou 200 reais?

- ```
SELECT cod_med, SUM(valor)
FROM atendimento
GROUP BY cod_med
HAVING SUM(valor) > 200;
```



GROUP BY e HAVING

9. Média do valor dos exames feitos entre um período cujo a média seja menor que 200?

- ```
SELECT dt_exame, AVG(valor)
FROM exame
WHERE dt_exame BETWEEN '30/03/11' AND '02/04/11'
GROUP BY dt_exame
HAVING AVG(valor) < 200;
```

10. Qual o nome do paciente que foi atendido mais da uma vezes?

- ```
SELECT p.nome, COUNT(*)
FROM paciente p, atendimento at
WHERE p.cod_pac= at.cod_pac
GROUP BY p.nome
HAVING COUNT(*)>=2;
```



GROUP BY e HAVING

11. Exiba o diagnóstico do atendimento e o mais baixo valor de uma consulta. Exclua qualquer atendimento onde o diagnóstico não seja conhecido. Exclua qualquer grupo onde o preço máximo é maior que 100 reais. Ordene o resultado em ordem decrescente de preço?
- ```
SELECT diagnostico, MIN(Valor)
FROM atendimento
WHERE diagnostico IS NOT NULL
GROUP BY diagnostico
HAVING MIN(valor) < 100
ORDER BY MIN(valor) desc;
```



# Bibliografias

- HEUSER, Carlos Alberto, “Projeto de Banco de Dados”. 4ª Edição (Série Livros Didáticos Nº 4), Porto Alegre: Instituto de Informática da UFRGS, 2001.
- RAMIREZ, Elmasri, S. Navathe a “Sistemas de Banco de Dados”. 4ª edição, São Paulo: Editora Pearson, 2005.
- SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S., “Sistemas de Banco de Dados”. Tradução da 5ª edição, Rio de Janeiro: Editora Campus, 2006.
- SILVA, Robson Soares, “Oracle 10g Express Edition: Guia de Instalação, Configuração e Administração”. 1ª edição, São Paulo: Érica, 2007.