2.3. Banco de dados relacionais - SQL COUNT,AVG, SUM,MAX e MIN

06/10/2023

SQL COUNT

A função **SQL COUNT** é importante para obtermos o total de registros que correspondam a uma determinada condição feita em uma instrução SQL. Além dessa função, a linguagem SQL oferece outras opções de funções agregadas que são utilizadas de diferentes formas, entre elas: AVG(), SUM(), MAX() e MIN().

A função **SQL COUNT** é utilizada para determinar o valor total de registros que atenda a uma condição específica.

A sintaxe da função SQL COUNT é:

```
SELECT  COUNT(nome_coluna)
FROM nome_tabela
WHERE condição
```

Exemplo de uso

```sql
CREATE DATABASE IF NOT EXISTS Escola;
USE Escola;
CREATE TABLE IF NOT EXISTS Pessoa(
   pessoa_id INT AUTO_INCREMENT PRIMARY KEY,
   nome_pessoa VARCHAR(255) NOT NULL,
   idade int,
   cidade VARCHAR(255),
   nota1 decimal(10,2),
   nota2 decimal(10,2),
   valor_mensalidade decimal(10,2),
   valor_desconto decimal(10,2)

   );

INSERT INTO Pessoa
VALUES
   (1, 'João', 23, 'São Paulo', 8.2, 7.0, 289.90, 50.00),
   (2, 'Maria', 35, 'Belo Horizonte', 6.2, 9.5, 375.50, 30.00),
   (3, 'Cláudia',25, 'São Paulo', 8.0, 9.0, 188.00, 58.00),
   (4, 'Pedro', null, 'Rio de Janeiro', 8.0, 9.2, 250.00, 42.00),
   (5, 'Sérgio', 34, null, 9.0, 6.5, 250.00, null),
   (6, 'Solange', 18, null, 8.0, 7.5, null, 30.00),
   (7, 'Rodrigo', 28, 'Campinas', null, null, null, null),
   (8, 'Marlene', 30, 'Campinas', 9.0, 8.0, null, null)
```

Vamos contar o total de registros na tabela “**Pessoa**”. Veja a instrução SQL:

```sql
SELECT COUNT(*) AS total_registros FROM pessoa
```

Resultado:
total_registros
8

Também podemos indicar um campo específico. Veja o exemplo a seguir:

```
SELECT COUNT(cidade) AS total_cidade FROM pessoa
```

Resultado:
total_cidade
6

Na seleção acima, selecionamos o total de registros em que o campo cidade está preenchido. É importante dizer que **a função COUNT() considera apenas valores válidos, ou seja, não conta os valores nulos**. Além disso, **ela não faz distinção se o conteúdo do campo é repetido ou não**. Por isso, a quantidade de cidades é de 6, apesar de existirem registros com cidades de mesmo nome.

SQL COUNT com GROUP BY

Podemos utilizar a função **COUNT()** para descobrir a quantidade de registros que pertença a um determinado grupo de dados. Veja a instrução SQL a seguir:

```
SELECT
   cidade,
   COUNT(*) AS total_por_cidade
FROM pessoa
GROUP BY cidade
```

Resultado:
Cidade / total_por_cidade
Null / 2
Belo Horizonte / 1
Campinas / 2
Rio de Janeiro / 1
São Paulo / 2

Perceba que temos duas cidades com o conteúdo igual a NULL entre os resultados listados. Isso acontece porque **a cláusula GROUP BY considera os valores nulos**

como válidos para o agrupamento

.

Se quisermos excluir esses valores:

```
SELECT cidade, COUNT(*) AS total_por_cidade
FROM pessoa
WHERE cidade IS NOT NULL
GROUP BY cidade
```

Resultado:
cidade / total_por_cidade
Belo Horizonte / 1
Campinas / 2
Rio de Janeiro / 1
São Paulo / 2

SQL COUNT com ORDER BY

A cláusula ORDER BY é utilizada para ordenar o resultado de acordo com um critério específico.

.

```
SELECT
cidade,
count() AS total_por_cidade
FROM pessoa
WHERE cidade IS NOT NULL
GROUP BY cidade
ORDER BY count()
```

Resultado:
cidade / total_por_cidade
Belo Horizonte / 1
Rio de Janeiro / 1
São Paulo / 2
Campinas / 2

SQL COUNT com HAVING

A cláusula HAVING é utilizada para adicionarmos uma condição na seleção de dados que é feita com o uso de funções agregadas, como a COUNT(), pois a cláusula WHERE não pode ser utilizada com funções desse tipo.

```
SELECT
cidade,
count() AS total_por_cidade
FROM pessoa
GROUP BY cidade
HAVING COUNT() > 1
```

Resultado:
cidade / total_por_cidade
Null / 2
Campinas / 2
São Paulo / 2

Perceba que temos os valores nulos listados como resultado. Isso porque utilizamos o caractere “*” na função COUNT() na cláusula HAVING. Para eliminar esses dados, podemos adicionar a cláusula WHERE ou indicar o campo cidade na função COUNT().

```
SELECT
cidade,
count(*) AS total_por_cidade
FROM pessoa
GROUP BY cidade
HAVING COUNT(cidade) > 1
```

Resultado:
cidade / total_por_cidade
Campinas / 2
São Paulo / 2

SQL AVG

A função SQL AVG() é utilizada para retornar o valor médio de um grupo de registros selecionados com a cláusula SELECT. Ela só pode ser aplicada em campos que contenham valores numéricos.

A sintaxe da função AVG() é:

```
SELECT AVG( [ALL/DISTINCT] nome_coluna ou expressão)
FROM nome_tabela
WHERE condição
```

Exemplos de como utilizar a função AVG().

```
SELECT
cidade,
AVG(nota1) AS nota_media
FROM pessoa
GROUP BY cidade
```

Resultado:
cidade / nota_media
NULL/ 8.500000
Belo Horizonte / 6.200000
Campinas / 9.000000
Rio de Janeiro / 8.000000
São Paulo / 8.100000

Como mencionamos, a função AVG() ignora os campos com valores nulos. Por isso, no resultado apresentado, o valor calculado da média para a cidade de Campinas é igual a 9.0 apesar de termos dois registros para essa cidade.

Entretanto, temos uma cidade listada com o valor nulo e com o cálculo da média realizado para ela. Nesse caso, a média foi calculada, pois não há valores nulos para o campo “nota1” e sim, para o campo “cidade”, que é utilizado pela cláusula GROUP BY — que considera valores nulos para realizar agrupamentos. Portanto, se quisermos excluir os valores nulos, devemos adicionar essa condição na instrução SQL.

A função AVG() também pode ser aplicada com uma expressão.

```
SELECT
cidade,
AVG((nota1 + nota2)/2) AS notas_media
FROM pessoa
GROUP BY cidade
```

Resultado:
cidade / notas_media
Null / 7.7500000000
Belo Horizonte / 7.8500000000
Campinas / 8.5000000000
Rio de Janeiro / 8.6000000000
São Paulo / 8.0500000000

SQL SUM

A sintaxe da função SUM() é:

```
SELECT SUM ([ALL / DISTINCT] nome_campo ou expressão)
FROM nome_tabela
WHERE condição
```

Somaremos os valores correspondentes às mensalidades agrupados por cidade.

```
SELECT
cidade,
SUM(valor_mensalidade) AS total_mensalidade
FROM pessoa
GROUP BY cidade
```

Resultado:
cidade / total_mensalidade
Null / 250.00
Belo Horizonte / 375.50
Campinas / Null
Rio de Janeiro / 250.00
São Paulo / 477.90

Perceba que **a função SUM() não considera os valores nulos ao fazer a somatória**. Além disso, perceba o que acontece na cidade de Campinas no nosso exemplo: **se todos os campos tiverem com valores iguais a nulo**, **o resultado será igual a nulo.**

A função SUM() também pode ser utilizada com dois ou mais campos do mesmo registro. Entretanto, se eles não fizerem parte de uma expressão, devem ser chamados de forma individual.

```
SELECT
cidade,
SUM(valor_mensalidade) AS total_mensalidade,
SUM(valor_desconto) AS total_desconto
FROM pessoa
GROUP BY cidade
```

Resultado:
cidade / total_mensalidade / total_desconto
Null / 250.00 / 30.00
Belo Horizonte / 375.50 / 30.00
Campinas / Null / Null
Rio de Janeiro / 250.00 / 42.00
São Paulo / 477.90 / 108.00

Importante: **A função SUM() somente deve ser utilizada com campos que contenham valores numéricos**.

SQL MAX

A sintaxe da função MAX() é:

```
SELECT MAX(nome_coluna)
FROM nome_tabela
WHERE condição
```

A função MAX() retorna o maior valor de um determinado campo comparando todos os dados da tabela.

```
SELECT MAX(idade) AS idade_maxima FROM pessoa;
```

Resultado:
idade_maxima
35

Também podemos utilizar essa função com o agrupamento de registros.

```
SELECT
cidade,
MAX(idade) AS idade_maxima
FROM pessoa
WHERE CIDADE IS NOT NULL
GROUP BY cidade
```

Resultado:
cidade / idade-maxima
Belo Horizonte / 35
Campinas / 30
Rio de Janeiro / Null
São Paulo / 25

A função MAX() ignora valores nulos. Além disso, ela retorna nulo caso não exista valores a serem considerados.

SQL MIN

A sintaxe da função MIN() é:

```
SELECT MIN(nome_coluna)
FROM nome_tabela
WHERE condição
```

A função MIX() retorna o menor valor de um determinado campo. Se nenhuma condição for definida, todos os dados da tabela serão comparados.

```
SELECT MIN(idade) AS idade_minima FROM pessoa;
```

Resultado:
idade_minima
18

Também podemos utilizar essa função com o agrupamento de registros.

```
SELECT
cidade,
MIN(idade) AS idade_minima
FROM pessoa
WHERE CIDADE IS NOT NULL
GROUP BY cidade
```

Resultado:
cidade / idade-minima
Belo Horizonte / 35
Campinas / 28
Rio de Janeiro / Null
São Paulo / 23

A função MIN() é semelhante à MAX(), porém, com a classificação ao contrário. Portanto, as recomendações para as duas são as mesmas. Vale ressaltar que é sempre importante realizar testes para conferir se as instruções SQL retornam os valores esperados.