Funções de agregação

Certified Developer
The Ultimate Tech Degree

DigitalHouse > Coding School

Anotações

“ As funções de agregação **realizam cálculos** sobre um conjunto de dados e **retornam um único resultado.**
Com exceção de **COUNT**, as funções de agregação irão **ignorar** os valores **NULL.**”

COUNT Sintaxe

Ele retornará o número de linhas / registro que atendem aos critérios.

```sql
SELECT COUNT(*) FROM movies;
```

Devolverá a quantidade de registros na tabela movies

```sql
SELECT COUNT(id) AS total FROM movies WHERE genre_id=3;
```

Devolverá a quantidade de filmes da tabela movies com genre_id 3 em uma coluna chamada total

AVG, SUM sintaxes

AVG (média): Retornará a média de uma coluna com valores numéricos.

SUM (soma): Retornará a soma de uma coluna com valores numéricos.

```sql
SELECT AVG(rating) FROM movies;
```

Ele retornará a classificação média dos filmes na tabela de filmes.

```sql
SELECT SUM(length) FROM movies;
```

Ele retornará a soma das durações dos filmes na tabela de filmes.

MIN, MAX sintaxes

MIN retornará o valor mínimo de uma coluna com valores numéricos.

MAX retornará o valor máximo de uma coluna.

```SQL
SELECT MIN(rating) FROM movies;
```

Ele retornará a classificação do filme menos classificado.

```SQL
SELECT MAX(length) FROM movies;
```

Ele retornará a classificação do filme com melhor classificação.

Group By

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Group By Sintaxe

GROUP BY é usado para **agrupar os registros** da tabela resultantes de uma consulta por uma ou mais colunas.

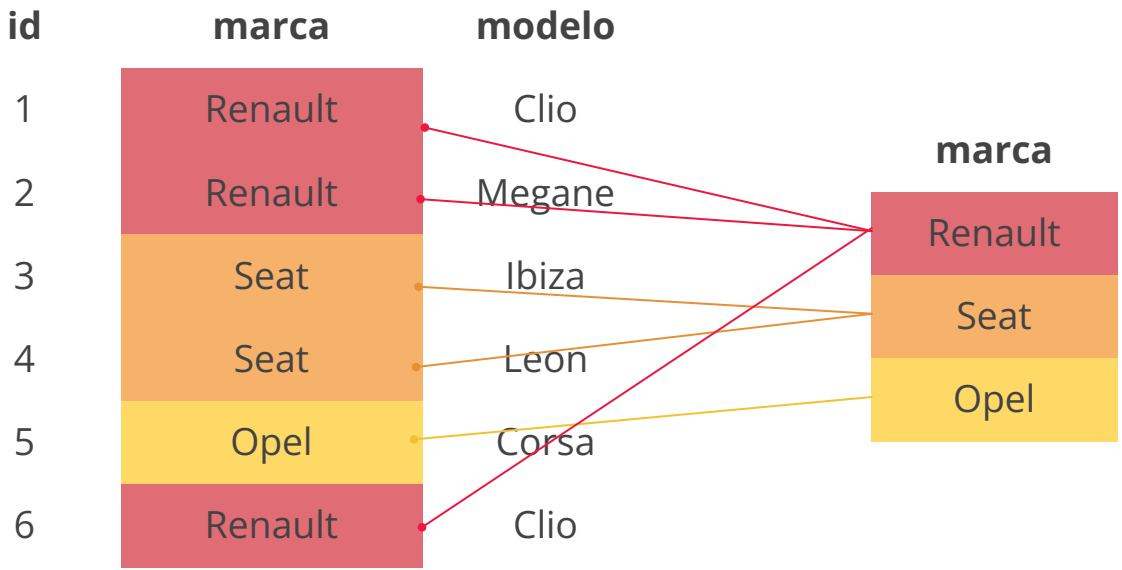
SQL

```sql
SELECT coluna_1
FROM nome_tabela
WHERE condição
GROUP BY coluna_1;
```

Group By Exemplo

```sql
SELECT marca,
FROM modelo
GROUP BY marca;
```

Group By Exemplo



Group By Sintaxe

Como **GROUP BY** agrupa as informações, perdemos o detalhe de cada uma das linhas. Em outras palavras, não estamos mais interessados no valor de cada linha, mas sim em um resultado consolidado entre todas as linhas.

Consulta:

```sql
SELECT id,marca
FROM modelo
GROUP BY marca;
```

Isso nos daria um erro. Se agruparmos os dados por marca, não podemos mais solicitar o campo id.

Group By Sintaxe

Portanto, ao usar **GROUP BY**, nos campos que aparecem como resultado do **SELECT** podemos apenas indicar:

- Dados agrupados
- Funções de agregação

Vejamos alguns exemplos ...

Group By Sintaxe

SQL

```sql
SELECT marca, MAX(preço)
FROM modelo
GROUP BY marca;
```

SQL

```sql
SELECT gênero.nome, AVG(duração)
FROM filmes
INNER JOIN gêneros ON gêneros.id = gênero_id
GROUP BY gênero.nome;
```

Having

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

HAVING Sintaxe

Ele cumpre a mesma função de **WHERE**, ao contrário de **HAVING** sendo usado em conjunto com as **funções de agregação** para filtrar **dados agregados.**

SQL

```sql
SELECT coluna_1
FROM nome_tabela
WHERE condição
GROUP BY coluna_1
HAVING condição_grupo
ORDER BY coluna_1;
```

HAVING Sintaxe

Esta consulta retornará o número de clientes por país (agrupados por país). Apenas os países com **pelo menos** 3 clientes serão incluídos no resultado.

```sql
SELECT país, COUNT(clienteId)
FROM clientes
GROUP BY país
HAVING COUNT(clienteId)>3;
```

Joins

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Por que usar JOINS?

Além de fazer consultas em uma tabela ou em muitas tabelas por meio de **referência de tabela**, também é possível e necessário consultar **diferentes tabelas** e unir esses resultados com **JOINS.**

Embora tenham a mesma função que a **referência da tabela**, o **JOINS**:

- Eles fornecem certas flexibilidades adicionais.
- Sua sintaxe é muito mais usada.
- Apresentam uma melhor performance.

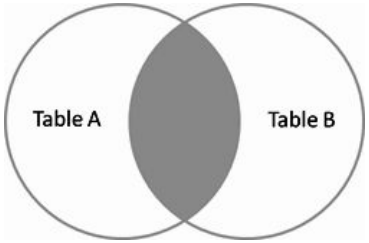
INNER JOIN

O **INNER JOIN** fará um **cruzamento** entre duas tabelas. Se cruzássemos as tabelas de clientes e vendas e houvesse um cliente sem vendas, o INNER JOIN **não traria** esse cliente como resultado.

CLIENTES

id	nome	sobrenome
1	João	Silva
2	Clara	Martins
3	Marta	Santos

INNER JOIN



VENDAS

id	cliente_id	data
1	2	12/03/2019
2	2	22/08/2019
3	1	04/09/2019

Criando um INNER JOIN

Antes escrevíamos:

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes, vendas
```

Agora escrevemos:

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes
INNER JOIN vendas
```

Anotações

“Embora já tenhamos dado o primeiro passo, que é **cruzar** as duas tabelas, ainda precisamos esclarecer **onde** fica esse cruzamento.
Logo, qual **chave primária (PK)** será cruzada com qual **chave estrangeira (FK).**”

Criando um INNER JOIN (cont.)

A sintaxe **JOIN** não usa **WHERE**, mas requer a palavra **ON**. Neste, indicaremos o **filtro** necessário para executar o processo.

Logo, o que escrevíamos no **WHERE**, agora vamos escrever no **ON**.

SQL

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes
INNER JOIN vendas
ON clientes.id = vendas.cliente_id
```


DigitalHouse>
Coding School