```
53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e
```

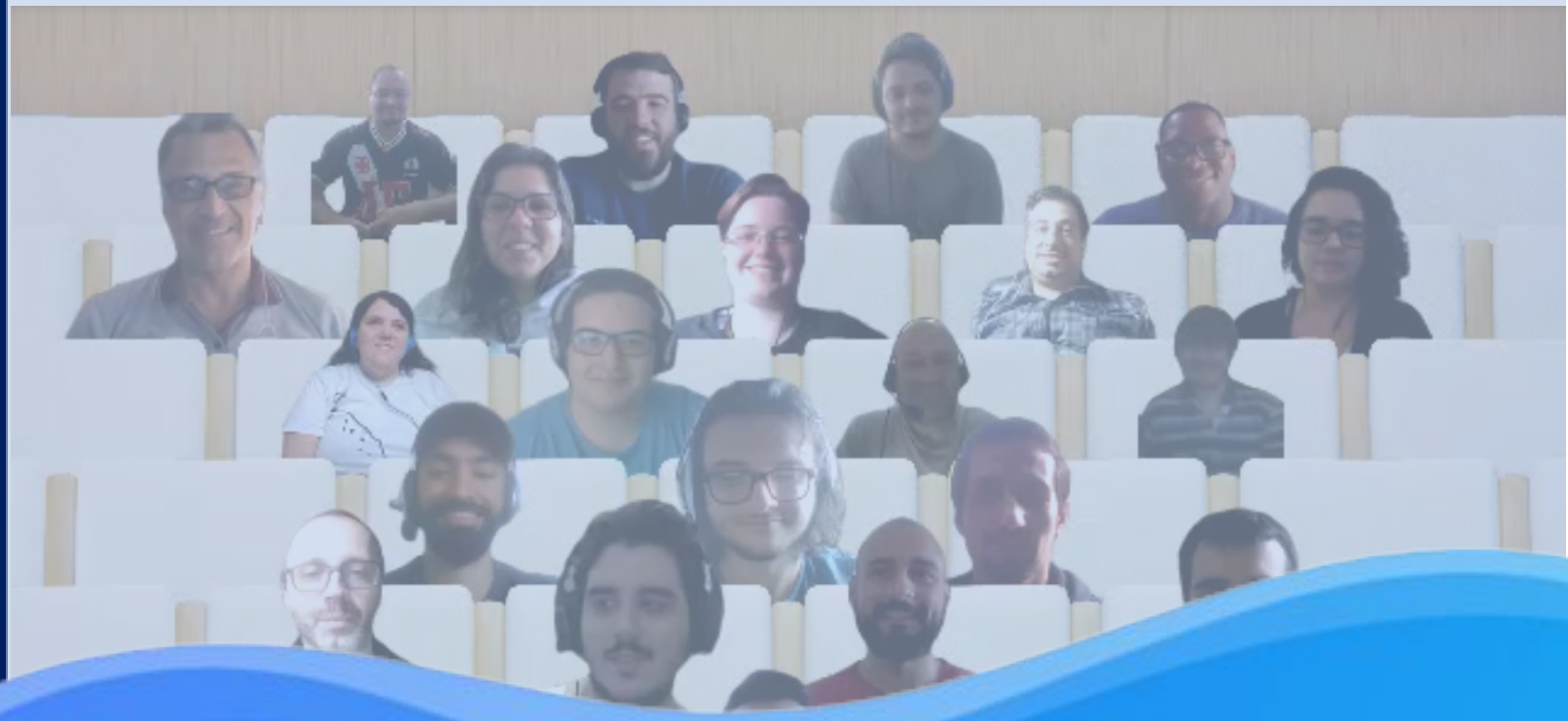
RESIDÊNCIA DE SOFTWARE

CAPACITAR
TREINAR
EMPREGAR
TRANSFORMAR

Firjan SENAI

serratec
Parque Tecnológico Região Serrana

Banco de Dados



Aula 5
Agregação

Roni Schanuel
24-08-2022

Funções de Agregação

São funções nativas que podem ser usadas para agregar um conjunto de valores em um único resultado.

COUNT - Retorna a quantidade de registros em uma tabela ignorando valores nulos. Retorna quantos produtos temos na tabela. No exemplo abaixo fazemos a inserção de um registro na tabela produto para ver a diferença entre o uso do **COUNT**.

```
INSERT INTO produto (nome, preco, quantidade_estoque) VALUES('Molho de Tomate', 2.95, 100)
```

```
SELECT COUNT(descricao) FROM produto;
```

Output | Explain | Messages | Notifications

count bigint
10

```
SELECT COUNT(*) FROM produto;
```

Output | Explain | Messages | Notific

count bigint
11

```
SELECT COUNT(*) FROM produto where preco > 7
```

Output | Explain | Messages | Notifications

count bigint
4

Funções de Agregação

MAX - Retorna o maior valor dos registros de uma tabela.

```
SELECT MAX(preco) FROM produto
```

MIN - Retorna o menor valor dos registros de uma tabela.

```
SELECT MIN(preco) FROM produto
```

SUM - Retorna a soma da coluna dos registros de uma tabela.

```
SELECT SUM(quantidade_estoque) AS SOMA FROM produto
```

a Output | Explain | Messages | Notifications

soma bigint
419

Funções de Agregação

AVG- Retorna a média de uma coluna da tabela.

```
SELECT ROUND(AVG(preco),2) AS MEDIA FROM produto
```

a Output | Explain | Messages | Notifications

media numeric	
5.65	

```
SELECT ROUND(AVG(preco),2) AS MEDIA FROM produto
WHERE nome = 'Feijão'
```

a Output | Explain | Messages | Notifications

media numeric	
8.50	

GROUP BY

Utilizada em conjunto com funções de agregação para o agrupamento do resultado por uma ou mais colunas.

Diferente da função SUM que retorna todas quantidades de forma isolada com o **GROUP BY** temos o total por nome do produto conforme exemplo

```
SELECT nome, sum(quantidade_estoque) FROM produto
GROUP BY nome ORDER BY nome;
```

ta Output | Explain | Messages | Notifications

nome character varying (40)	sum bigint
Arroz	44
Atum	42
Creme de Leite	15
Farinha de Trigo	8
Feijão	60
Leite Condensado	40
Macarrão	10
Molho de Tomate	100
Sal	100

GROUP BY

No exemplo abaixo estamos contanto o total de itens e depois utilizamos o group by para totalizar por cada item.

```
select COUNT (*) as total from produto
```

lts 1

:t COUNT (*) as total from produto | Enter a SQL expre

total
11

```
select nome, count(nome) from produto
group by nome order by nome
```

uto 1

:t nome, count(nome) from produto group by nc

nome	count
Arroz	2
Atum	1
Creme de Leite	1
Farinha de Trigo	1
Feijão	2
Leite Condensado	1
Macarrão	1
Molho de Tomate	1
Sal	1

HAVING

Serve para filtrar o resultado do group by. No exemplo abaixo estamos agrupando por nome do produto e fazendo um filtro no agrupamento para soma de quantidade em estoque maior que 50.

```
select nome, sum(quantidade_estoque)from produto
group by nome
having sum(quantidade_estoque) > 50 order by nome;
```

duto 1

ct nome, sum(quantidade_estoque | Enter a SQL expression to filter resul

nome	sum
Feijão	60
Molho de Tomate	100
Sal	100

HAVING

No exemplo abaixo a agregação é por nome do produto e somente da descrição(marca) carreteiro e no resultado filtramos com o having para os produtos com quantidade em estoque maior que 10

sem having

```
select nome, sum(quantidade_estoque) from produto
where descricao = 'Carreteiro'
group by nome
```

uto 1

:t nome, sum(quantidade_estoque) from produt Enter a SQL expres

nome	sum
Arroz	4
Feijão	30

com having

```
select nome, sum(quantidade_estoque) from produto
where descricao = 'Carreteiro'
group by nome
having sum(quantidade_estoque) > 10
```

uto 1

:t nome, sum(quantidade_estoque Enter a SQL expression to filter

nome	sum
Feijão	30

Vamos inserir mais dados em nossas tabelas para os exemplos a seguir

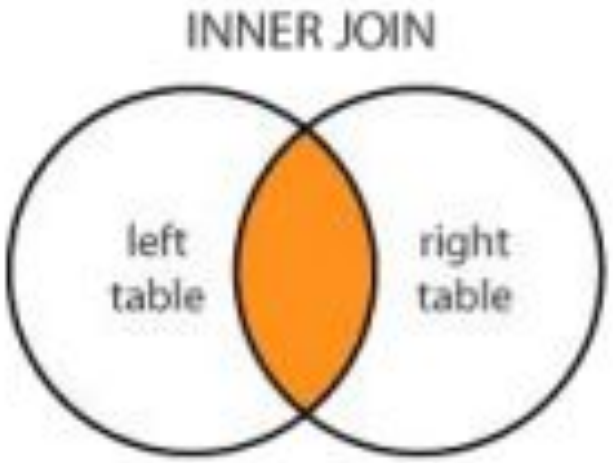
```
INSERT INTO aula.produto (nome, descricao, preco, quantidade_estoque,codigo_categoria) VALUES('Creme de Leite','Itambé',2.8,25,null);
INSERT INTO aula.produto (nome, descricao, preco, quantidade_estoque,codigo_categoria) VALUES('Arroz','Princesa',6.5,44,null);
INSERT INTO aula.produto (nome, descricao, preco, quantidade_estoque,codigo_categoria) VALUES('Colírio','EMS',22.5,30,null);
```

Junções
Muitas vezes as consultas precisam recuperar dados de tabelas diferentes, por isso, precisamos definir critérios de junções para obter estes dados.

Tipos de junções
INNER JOIN
Retorna todas as linhas quando há pelo menos um registro correspondente em ambas as tabelas.

OUTER JOIN
Retorna linhas mesmo quando não houver pelo uma correspondência em uma das tabelas. O OUTER JOIN se sub divide em LEFT JOIN, RIGHT JOIN e CROSS JOIN.

INNER JOIN




```sql
SELECT * FROM aula.produto p
    INNER JOIN aula.categoria c
    on p.codigo_categoria = c.codigo_categoria
```

uto(+) 1

CT * FROM aula.produto p INNER JOIN aula.categc | Enter a SQL expression to filter results (use Ctrl+Space)

codigo_produto	nome	descricao	preco	quantidade_estoque	codigo_categoria	codigo_categoria	descricao
1	Arroz	Tio João	7.5	40	2	2	Alimentos
2	Feijão	Carreteiro	8.5	30	2	2	Alimentos
3	Feijão	Granfino	8.5	30	2	2	Alimentos
4	Macarrão	Adria	6.65	10	2	2	Alimentos
5	Farinha de Trigo	Boa Sorte	2.5	8	2	2	Alimentos
6	Sal	Cisne	2.5	100	2	2	Alimentos
7	Atum	Gomes da Costa	6.8	42	2	2	Alimentos
8	Leite Condensado	Nestle	5.9	40	2	2	Alimentos
9	Creme de Leite	Pirancajuba	2.8	15	2	2	Alimentos
10	Arroz	Carreteiro	7.5	4	2	2	Alimentos
11	Rinosoro Jet	Neoquímica	17.5	4	1	1	Eletrônicos
12	Carne	[NULL]	40	200	2	2	Alimentos

INNER JOIN

Na imagem abaixo temos outra forma de realizar a junção.

```sql
select * from aula.produto p, aula.categoria c
        where c.codigo_categoria = p.codigo_categoria ;
```

uto(+) 1

:t * from aula.produto p, aula.categoria c where c.c | Enter a SQL expression to filter results (use Ctrl+Space)

codigo_produto	nome	descricao	preco	quantidade_estoque	codigo_categoria	codigo_categoria	descricao
1	Arroz	Tio João	7.5	40	2	2	Alimentos
2	Feijão	Carreteiro	8.5	30	2	2	Alimentos
3	Feijão	Granfino	8.5	30	2	2	Alimentos
4	Macarrão	Adria	6.65	10	2	2	Alimentos
5	Farinha de Trigo	Boa Sorte	2.5	8	2	2	Alimentos
6	Sal	Cisne	2.5	100	2	2	Alimentos
7	Atum	Gomes da Costa	6.8	42	2	2	Alimentos
8	Leite Condensado	Nestle	5.9	40	2	2	Alimentos
9	Creme de Leite	Pirancajuba	2.8	15	2	2	Alimentos
10	Arroz	Carreteiro	7.5	4	2	2	Alimentos
11	Rinosoro Jet	Neoquímica	17.5	4	1	1	Eletrônicos
12	Carne	[NULL]	40	200	2	2	Alimentos

LEFT JOIN

Retorna todas as linhas da tabela à esquerda, mesmo não havendo nenhuma correspondência na tabela à direita. No inner join só retornamos os valores correpondentes nos dois lados da relação, mas no LEFT JOIN retornamos também os produtos que não tem uma categoria definida.

```sql
SELECT * FROM aula.produto p
    LEFT JOIN aula.categoria c
    on p.codigo_categoria = c.codigo_categoria;
```

uto(+) 1

CT * FROM aula.produto p LEFT JOIN aula.categori

codigo_produto	nome	descricao	preco	quantidade_estoque	codigo_categoria	codigo_categoria	descricao
1	Arroz	Tio João	7.5	40	2	2	Alimentos
2	Feijão	Carreteiro	8.5	30	2	2	Alimentos
3	Feijão	Granfino	8.5	30	2	2	Alimentos
4	Macarrão	Adria	6.65	10	2	2	Alimentos
5	Farinha de Trigo	Boa Sorte	2.5	8	2	2	Alimentos
6	Sal	Cisne	2.5	100	2	2	Alimentos
7	Atum	Gomes da Costa	6.8	42	2	2	Alimentos
8	Leite Condensado	Nestle	5.9	40	2	2	Alimentos
9	Creme de Leite	Pirancajuba	2.8	15	2	2	Alimentos
10	Arroz	Carreteiro	7.5	4	2	2	Alimentos
11	Rinosoro Jet	Neoquímica	17.5	4	1	1	Eletrônicos
12	Carne	[NULL]	40	200	2	2	Alimentos
13	Creme de Leite	Itambé	2.8	25	[NULL]	[NULL]	[NULL]
14	Arroz	Princesa	6.5	44	[NULL]	[NULL]	[NULL]
15	Colírio	EMS	22.5	30	[NULL]	[NULL]	[NULL]

RIGHT JOIN

Retorna todas as linhas da tabela à direita, mesmo se não houver nenhuma correspondência na tabela à esquerda.

```sql
SELECT * FROM aula.produto p
    RIGHT JOIN aula.categoria c
    on p.codigo_categoria = c.codigo_categoria;
```

uto(+) 1

CT * FROM aula.produto p RIGHT JOIN aula.categc | Enter a SQL expression to filter results (use Ctrl+Space)

codigo_produto	nome	descricao	preco	quantidade_estoque	codigo_categoria	codigo_categoria	descricao
1	Arroz	Tio João	7.5	40	2	2	Alimentos
2	Feijão	Carreteiro	8.5	30	2	2	Alimentos
3	Feijão	Granfino	8.5	30	2	2	Alimentos
4	Macarrão	Adria	6.65	10	2	2	Alimentos
5	Farinha de Trigo	Boa Sorte	2.5	8	2	2	Alimentos
6	Sal	Cisne	2.5	100	2	2	Alimentos
7	Atum	Gomes da Costa	6.8	42	2	2	Alimentos
8	Leite Condensado	Nestle	5.9	40	2	2	Alimentos
9	Creme de Leite	Pirancajuba	2.8	15	2	2	Alimentos
10	Arroz	Carreteiro	7.5	4	2	2	Alimentos
11	Rinosoro Jet	Neoquímica	17.5	4	1	1	Eletrônicos
12	Carne	[NULL]	40	200	2	2	Alimentos
[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	5	Brinquedos
[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	4	Farmácia