

INFORMAÇÕES

MÓDULO	AULA	INSTRUTOR
01 Fundamentos de Banco de Dados	05 - Funções Avançadas e Window Functions	Matheus Laureto

CONTEÚDO









1. Revisão da Aula Anterior

- **Subconsultas (ou subqueries):**

- Definição: consultas dentro de outras colunas;
- Uso: buscar valores já processados para utilizar na consulta principal;
- Execução: as subconsultas sempre executam primeiro e, seu resultado é utilizado na query principal;
- Ex: funcionários que ganham mais do que a média dos salários.

```
SELECT nome, salario
FROM funcionarios
WHERE salario > (SELECT AVG(salario) FROM funcionarios);
```

Output Messages Notifications

								SQL
nome		salario						
character varying (50)		numeric (12,2)						
João		4000.00						

- **CTEs – Common Table Expression:**

- Definição: criação e utilização de “tabelas temporárias” crias apenas para a execução da consulta (em tempo de execução);
- Vantagens:
 - melhor entendimento do código e fácil leitura;
 - reuso sem repetição de código;
 - fácil de testar e depurar os trechos distintos;
 - ótima performance.

- Ex: clientes que compraram mais de R\$3000 no total:

```
WITH total_por_cliente AS (  
    SELECT cliente_id, SUM(valor) AS total_gasto  
    FROM vendas  
    GROUP BY cliente_id  
)  
SELECT cliente_id, total_gasto  
FROM total_por_cliente  
WHERE total_gasto > 3000;
```

Output Messages Notifications

cliente_id	total_gasto
integer	numeric
3	3200.00
1	4899.00

2. FUNÇÕES NUMÉRICAS

- São funções que nos ajudam a trabalhar com números.
São utilizadas quando precisamos tratar valores antes de mostrar em relatórios, cálculos ou análises.

- **ROUND()**

Comando utilizado para arredondar valores:

```
SELECT ROUND(123.456, 2);
```

Output Messages Notifications









round
numeric
123.46

- **CEIL() / FLOOR()**

Comandos para arredondar para cima ou para baixo:

```
SELECT CEIL(4.1), FLOOR(4.9);
```

Output Messages Notifications








								SQL
ceil	numeric	lock	floor	numeric	lock			
	5			4				

- **ABS()**

Retorna valor absoluto:

```
SELECT ABS(-15);
```

Output Messages Notificatio









						
abs	integer	lock				
	15					

- **RANDOM()**

Retorna números aleatórios:

```
SELECT RANDOM();  
SELECT (RANDOM() * 10 + 1)::int;
```

Output Messages Notifications

								SQL
int4	integer	lock						
	3							

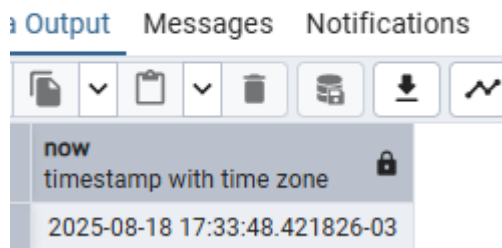
3. FUNÇÕES DE DATA E HORA

- Essas funções de data e hora servem para conseguir manipular e analisar informações temporais no banco de dados, ou seja, tudo o que envolve tempo, datas e períodos.

- **NOW()**

Função utilizada para retornar a data e hora do banco de dados:

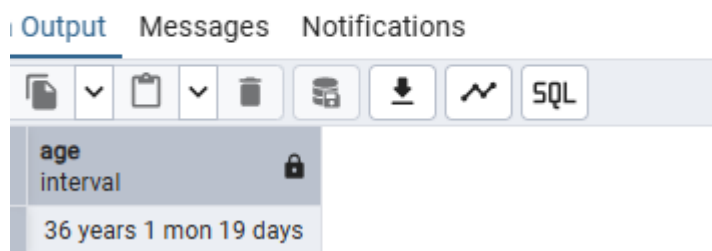
```
SELECT NOW();
```



- **AGE()**

Função que, ao passar data inicial e data final, calcula a diferença de tempo entre elas:

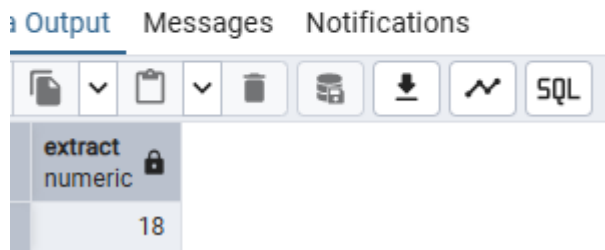
```
SELECT AGE('2025-08-18', '1989-06-29');
```



- **EXTRACT()**

Extrai partes específicas de uma data (ano, mês, dia, hora, minuto, segundo):

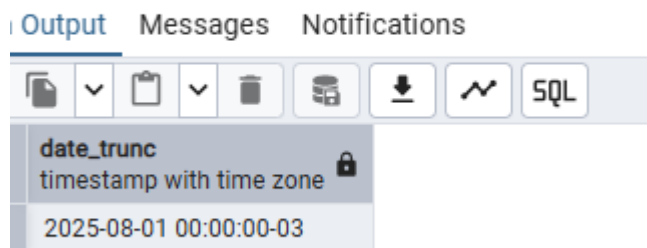
```
SELECT EXTRACT(YEAR FROM NOW());  
SELECT EXTRACT(MONTH FROM NOW());  
SELECT EXTRACT(DAY FROM NOW());
```



- **DATE_TRUNC()**

Trunca uma data para o início de um período:

```
SELECT DATE_TRUNC('month', NOW());
```



4. WINDOW FUNCTION (AGREGAÇÃO EM LINHA)

- São funções que calculam valores sobre um conjunto de linhas sem “agrupar” tudo em uma única linha.
- Diferente do **GROUP BY**, que resume os dados, o WINDOW FUNCTION mantém todas as linhas e adiciona uma coluna com o cálculo.
- Ex: total gasto por cliente sem perder os detalhes:

```
SELECT v.id,  
       v.cliente_id,  
       v.valor,  
       SUM(v.valor) OVER (PARTITION BY v.cliente_id) AS total_cliente  
FROM vendas v;
```

Output Messages Notifications

id [PK] integer	cliente_id integer	valor numeric (10,2)	total_cliente numeric
5	1	399.00	4899.00
1	1	4500.00	4899.00
2	2	150.00	150.00
3	3	3200.00	3200.00
4	4	200.00	200.00
6	5	120.00	120.00
7	6	250.00	250.00

Essa consulta faz o seguinte: lista todas as colunas colocados no SELECT, e no final, cria uma “janela”, fazendo a soma de VALOR sendo particionado por CLIENTE_ID, ou seja, no final de cada linha, agrupado por cada cliente, já é calculada a soma do total por cliente.

- Ranking de clientes:

```
✓ SELECT cliente_id,  
        SUM(valor) AS total,  
        RANK() OVER (ORDER BY SUM(valor) DESC) AS posicao  
FROM vendas  
GROUP BY cliente_id;
```

Output Messages Notifications

cliente_id integer	total numeric	posicao bigint
1	4899.00	1
3	3200.00	2
6	250.00	3
4	200.00	4
2	150.00	5
5	120.00	6

Nesse caso, a consulta faz um rankeamento, colocando em ordem dos clientes que tiveram maior gasto.

- Soma Acumulada:

```
SELECT data_venda,
       valor,
       SUM(valor) OVER (ORDER BY data_venda) AS acumulado
FROM vendas;
```

Output Messages Notifications

data_venda date	valor numeric (10,2)	acumulado numeric
2024-10-01	4500.00	4500.00
2024-10-02	150.00	4650.00
2024-10-05	3200.00	7850.00
2024-10-06	200.00	8050.00
2024-10-07	399.00	8449.00
2024-10-08	120.00	8569.00
2024-10-10	250.00	8819.00

Cada linha mostra o valor de venda acumulando com o registro anterior, ordenado por data de venda

5. Exercícios

1. Na tabela VENDAS, liste ID, VALOR e o valor da venda arredondando com duas casas decimais.
2. Na tabela VENDAS, liste ID, VALOR e o ano de cada venda.
3. Na tabela VENDAS, liste ID, DATA_VENDA e calcule quanto tempo já se passou de cada venda.
4. Na tabela VENDAS, liste cada venda mostrando também o total gasto pelo cliente (partition by).
5. Mostre um ranking dos clientes que mais gastaram, em ordem decrescente de total.