

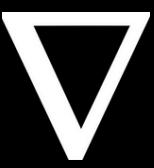
PostgreSQL

Aula 04



Presença

- Linktree: Presente na bio do nosso instagram
- Presença ficará disponível até 1 hora antes da próxima aula
- É necessário 70% de presença para obter o certificado



Presença



Séries Temporais

Tópicos Principais

1. Fundamentos da Manipulação de Dados Temporais
2. Agregação e Análise de Tendências
3. Técnicas Avançadas de Análise de Séries Temporais

Temporais



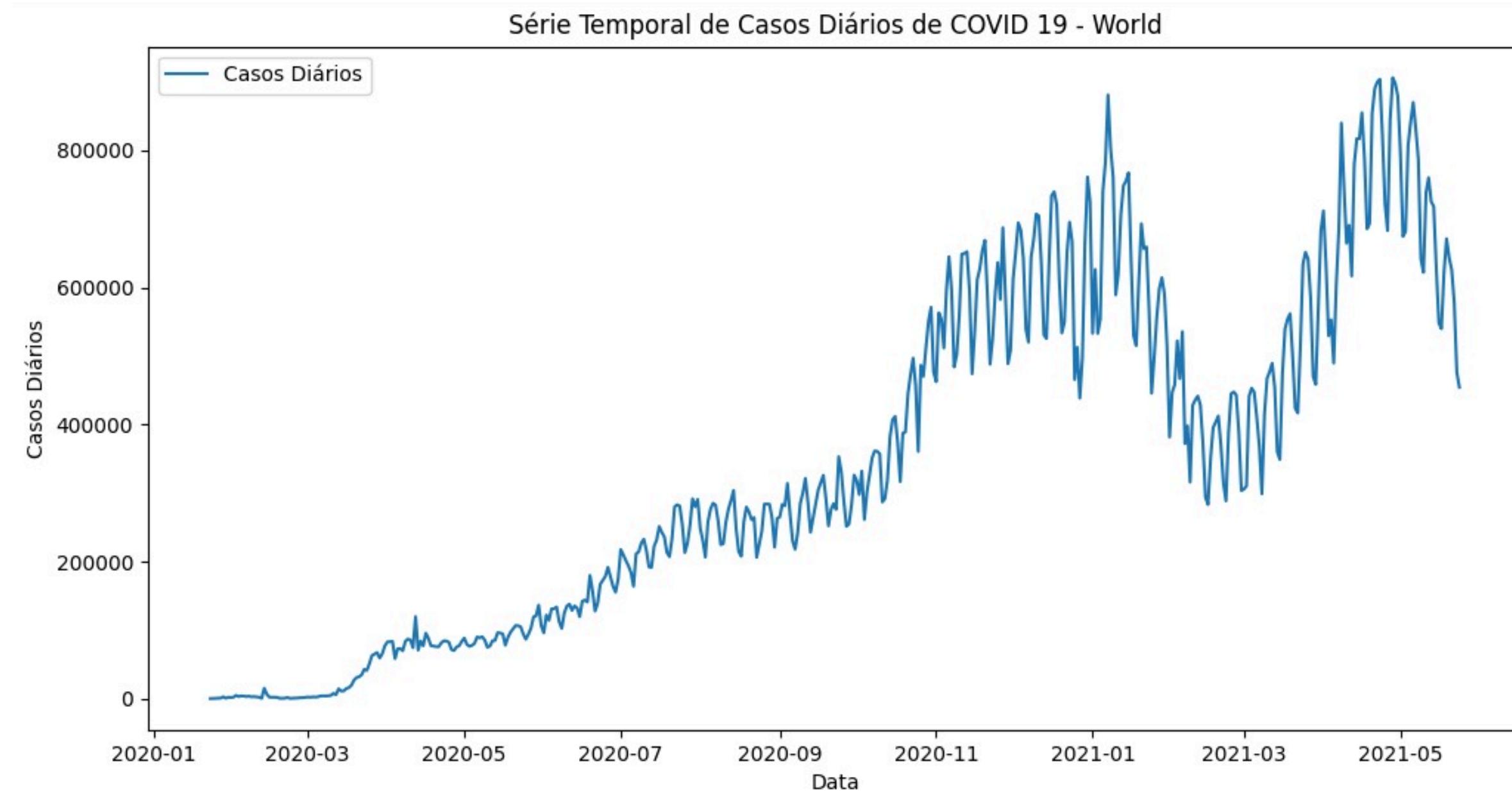
Introdução a séries temporais



Introdução a séries temporais

Uma série temporal é uma sequência de observações coletadas ao longo do tempo, geralmente em intervalos regulares, como dias, horas ou minutos. A ordem dos dados é essencial, pois o valor de um momento pode depender de valores anteriores.

Introdução a séries temporais





Tipos de Dados de Data e Hora no SQL

Tipos de Dados de Data e Hora no SQL

- DATE: armazena apenas a data (ano, mês, dia).
- TIME: armazena apenas a hora (hora, minuto, segundo).
- DATETIME / TIMESTAMP: armazena data e hora completas.

Tipos de Dados de Data e Hora no SQL

Boas práticas

- Sempre use TIMESTAMP se precisar de registro completo de data e hora.
- Armazene datas no formato UTC (Coordinated Universal Time), que é o padrão de tempo internacional usado como referência global, sem sofrer alterações por fusos horários ou horário de verão.



Funções Essenciais de Data e Hora

Funções Essenciais de Data e Hora

Extract: Extrai um componente de uma Data

```
1 ✓ SELECT EXTRACT(YEAR FROM datahora) AS ano,  
2   EXTRACT(MONTH FROM datahora) AS mes,  
3   EXTRACT(DAY FROM datahora) AS dia  
4   FROM vendas
```

Funções Essenciais de Data e Hora

EXTRACT: Extrai um componente de uma Data

```
1 ✓ SELECT EXTRACT(YEAR FROM datahora) AS ano,  
2   EXTRACT(MONTH FROM datahora) AS mes,  
3   EXTRACT(DAY FROM datahora) AS dia  
4   FROM vendas
```

Funções Essenciais de Data e Hora

DATE_PART: Similar ao **EXTRACT**

```
1 ✓ SELECT DATE_PART('hour', datahora) AS hora,  
2 SUM(vendas) AS total_vendas  
3 FROM vendas  
4 GROUP BY hora  
5 ORDER BY hora;
```

Funções Essenciais de Data e Hora

Obtenção de data ou horario atual

```
1  SELECT NOW() AS datahora_atual;  
2  SELECT CURRENT_DATE AS data_atual;  
3  SELECT CURRENT_TIMESTAMP AS timestamp_atual;
```



Manipulação e Aritmética com Datas

Manipulação Aritimética com datas

Na análise de séries temporais no PostgreSQL, frequentemente precisamos manipular datas: prever prazos adicionando intervalos, calcular o tempo decorrido entre eventos ou organizar registros em períodos bem definidos.

Manipulação Aritimética com datas

O que abordaremos:

- Adição e subtração de intervalos de tempo (INTERVAL).
- Cálculo da diferença entre duas datas (subtração direta ou uso de AGE).
- Truncamento de datas para o início do período: DATE_TRUNC.

Manipulação Aritimética com datas

Adição e Subtração de Intervalos

id_pedido	data_venda	data_entrega
1	2025-01-05 10:23:54	2025-01-10 15:00:00
2	2025-01-15 14:12:31	2025-01-20 09:30:00
3	2025-02-08 11:30:00	2025-02-18 16:45:00
4	2025-02-25 16:20:15	2025-02-28 10:00:00

Podemos calcular a entrega prevista (5 dias após a venda) usando:

```
1 ▾ SELECT
2   id_pedido,
3   data_venda,
4   data_venda + INTERVAL '5 days' AS entrega_prevista
5   FROM entregas;
```

Manipulação Aritimética com datas

Diferença entre Datas

id_pedido	data_venda	data_entrega
1	2025-01-05 10:23:54	2025-01-10 15:00:00
2	2025-01-15 14:12:31	2025-01-20 09:30:00
3	2025-02-08 11:30:00	2025-02-18 16:45:00
4	2025-02-25 16:20:15	2025-02-28 10:00:00

Para calcular quantos dias cada entrega levou:

```
1 ▾ SELECT
2   id_pedido,
3   EXTRACT(DAY FROM data_entrega - data_venda) AS dias_para_entrega
4   FROM entregas;
```

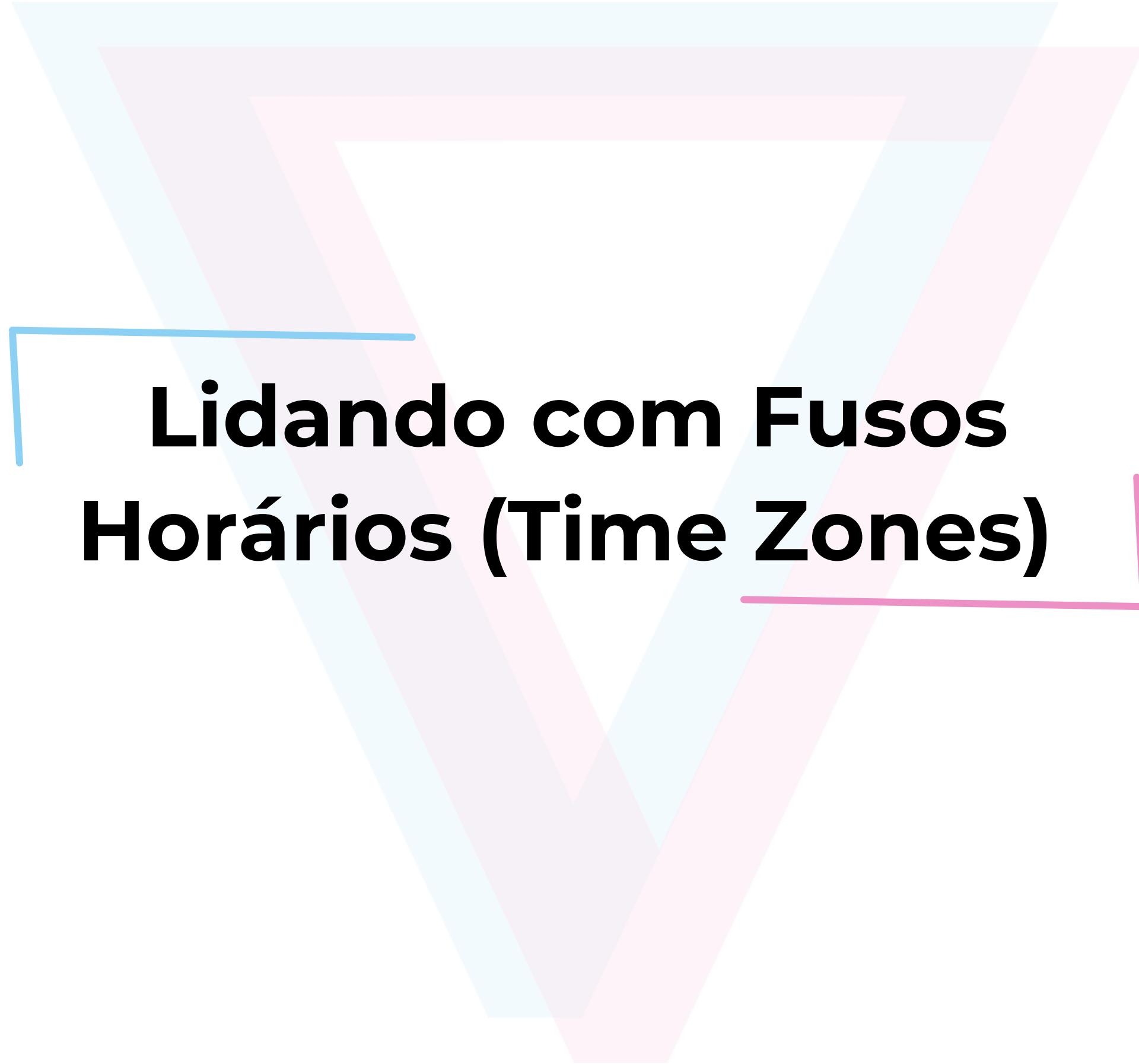
Manipulação Aritimética com datas

Truncamento de datas

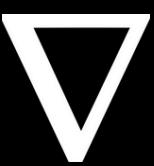
id_pedido	data_venda	data_entrega
1	2025-01-05 10:23:54	2025-01-10 15:00:00
2	2025-01-15 14:12:31	2025-01-20 09:30:00
3	2025-02-08 11:30:00	2025-02-18 16:45:00
4	2025-02-25 16:20:15	2025-02-28 10:00:00

Para agrupar pedidos por mês:

```
1 ▾ SELECT
2   DATE_TRUNC('month', data_venda) AS mes,
3   COUNT(*) AS pedidos
4   FROM entregas
5   GROUP BY mes
6   ORDER BY mes;
```



Lidando com Fusos Horários (Time Zones)



Lidando com Fusos Horários (Time Zones)

O que abordaremos:

- A importância da padronização de fusos horários (e.g., UTC)
- Funções para conversão entre diferentes fusos horários (AT TIME ZONE).

Lidando com Fusos Horários (Time Zones)

Em sistemas distribuídos, registros podem vir de diferentes regiões do mundo. Para evitar inconsistências, a prática recomendada é armazenar todas as datas no fuso horário UTC. A partir disso, podemos converter para o horário local sempre que necessário.

Lidando com Fusos Horários (Time Zones)

Tabela Exemplo:

id_acesso	data_acesso (UTC)
1	2025-01-05 15:00:00+00
2	2025-01-05 22:30:00+00
3	2025-01-06 01:15:00+00
4	2025-01-06 10:45:00+00

Se quisermos visualizar os horários no fuso de São Paulo:

```
1 ▾ SELECT
2   id_acesso,
3   data_acesso AT TIME ZONE 'UTC' AT TIME ZONE 'America/Sao_Paulo'
4   AS horario_local
5   FROM acessos;
```

Lidando com Fusos Horários (Time Zones)

Assim o resultado será:

id_acesso	horario_local (São Paulo)
1	2025-01-05 12:00:00
2	2025-01-05 19:30:00
3	2025-01-05 22:15:00
4	2025-01-06 07:45:00

Assim, conseguimos adaptar os relatórios para diferentes regiões sem comprometer a integridade da base original.



Agrupamento de Dados Temporais





Agrupamento de Séries Temporais

- Analisar dados temporais em intervalos de tempo
 - Dias
 - Semanas
 - Meses
- GROUP BY + Truncamento de datas
 - Utilização de funções de agregação de acordo com a análise

Agrupamento de Séries Temporais

id_venda	id_produto	data_venda	valor
1	101	2025-01-05 10:23:54	150.00
2	102	2025-01-15 14:12:31	75.50
3	103	2025-01-15 19:45:10	299.90
4	101	2025-01-28 08:55:00	150.00
5	201	2025-02-08 11:30:00	45.00
6	202	2025-02-18 21:05:49	199.99
7	102	2025-02-25 16:20:15	75.50
8	301	2025-03-02 12:00:00	89.90
9	101	2025-03-10 13:45:00	149.90
10	302	2025-03-20 09:10:25	550.00
11	201	2025-03-20 18:25:33	45.00
12	103	2025-03-29 22:15:00	299.90
13	401	2025-04-15 10:05:12	1200.00
14	402	2025-04-15 15:30:00	3300.00
15	101	2025-04-16 09:00:45	150.00

Quão bem foram as vendas ao longo dos 4 primeiros meses de 2025?

Agrupamento de Séries Temporais

id_venda	id_produto	data_venda	valor
1	101	2025-01-05 10:23:54	150.00
2	102	2025-01-15 14:12:31	75.50
3	103	2025-01-15 19:45:10	299.90
4	101	2025-01-28 08:55:00	150.00
5	201	2025-02-08 11:30:00	45.00
6	202	2025-02-18 21:05:49	199.99
7	102	2025-02-25 16:20:15	75.50
8	301	2025-03-02 12:00:00	89.90
9	101	2025-03-10 13:45:00	149.90
10	302	2025-03-20 09:10:25	550.00
11	201	2025-03-20 18:25:33	45.00
12	103	2025-03-29 22:15:00	299.90
13	401	2025-04-15 10:05:12	1200.00
14	402	2025-04-15 15:30:00	3300.00
15	101	2025-04-16 09:00:45	150.00

Ideia

2025-01-01 00:00:00

2025-02-01 00:00:00

2025-03-01 00:00:00

2025-04-01 00:00:00

Agrupamento de Séries Temporais

```
1  SELECT
2      DATE_TRUNC('month', data_venda) AS mes,
3      COUNT(id_venda) AS numero_de_vendas,
4      SUM(valor) AS receita_total,
5      AVG(valor) AS ticket_medio
6  FROM vendas
7  WHERE data_venda >= '2025-01-01' AND data_venda < '2025-05-01'
8  GROUP BY mes
9  ORDER BY mes;
```

mes	numero_de_vendas	receita_total	ticket_medio
2025-01-01 00:00:00	4	675.40	168.85
2025-02-01 00:00:00	3	320.49	106.83
2025-03-01 00:00:00	5	1134.70	226.94
2025-04-01 00:00:00	8	14405.89	1800.74

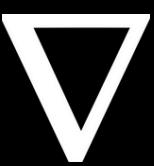


Agrupamento de Séries Temporais

- Geração de relatórios temporais
 - Útil para análise de tendências
 - Útil para geração de insights



Análise de Tendências Simples





Tendências Simples

- Análise da direção de evolução dos dados (crescimento e decrescimento)
 - Maior entendimento dos dados
- Exemplo das vendas
 - A partir de fevereiro, houve um grande crescimento de vendas, tendo um boom em abril

Tendências Simples - Acesso a Site

id_acesso	id_usuario	data_acesso				
1	123	2025-03-20 09:05:11		16	456	2025-03-23 16:40:10
2	456	2025-03-20 09:15:21		17	101	2025-03-23 20:55:00
3	789	2025-03-20 10:30:00		18	123	2025-03-24 09:00:00
4	123	2025-03-20 11:12:45		19	456	2025-03-24 09:10:00
5	101	2025-03-20 14:00:19		20	789	2025-03-24 10:15:20
6	456	2025-03-20 15:22:30		21	222	2025-03-24 11:30:00
7	123	2025-03-20 20:05:00		22	101	2025-03-24 14:20:50
8	456	2025-03-21 08:30:00		23	123	2025-03-24 15:00:00
9	789	2025-03-21 10:45:10		24	456	2025-03-24 16:45:10
10	222	2025-03-21 11:00:00		25	789	2025-03-25 08:55:00
11	123	2025-03-21 13:20:15		26	123	2025-03-25 09:30:00
12	456	2025-03-21 17:50:00		27	456	2025-03-25 10:00:00
13	789	2025-03-22 11:05:00		28	333	2025-03-25 12:10:15
14	333	2025-03-22 15:25:30		29	101	2025-03-25 14:40:00
15	123	2025-03-22 18:00:00		30	222	2025-03-25 15:30:45

Objetivo: Analisar os acessos ao site por dia

Tendências Simples - Acesso a Site

```
1  SELECT
2      EXTRACT(DOW FROM data_acesso) AS dia_da_semana,
3      TO_CHAR(data_acesso, 'Day') AS nome_do_dia,
4      COUNT(DISTINCT id_usuario) AS usuarios_ativos
5  FROM acessos_site
6  GROUP BY dia_da_semana, nome_do_dia
7  ORDER BY dia_da_semana;
```

EXTRACT (DOW FROM data_acesso)

- Retorna o dia da semana do Timestamp (0 para domingo até 6 para sábado)

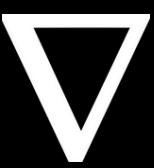
Tendências Simples - Acesso a Site

```
1  SELECT
2      EXTRACT(DOW FROM data_acesso) AS dia_da_semana,
3      TO_CHAR(data_acesso, 'Day') AS nome_do_dia,
4      COUNT(DISTINCT id_usuario) AS usuarios_ativos
5  FROM acessos_site
6  GROUP BY dia_da_semana, nome_do_dia
7  ORDER BY dia_da_semana;
```

dia_da_semana	nome_do_dia	usuarios_ativos
0	Sunday	2
1	Monday	5
2	Tuesday	6
4	Thursday	4
5	Friday	4
6	Saturday	3



Geração de Séries Temporais Completas



Séries Temporais Completas

Exemplo anteriores: Séries Temporais Contínuas

- Vendas: todos os meses presentes
- Acessos ao site: todos os dias presentes

Se houvesse dados faltantes, o GROUP BY não iria inserir esses no resultado, trazendo problemas:

- **Plotagem gráfica:** se utilizar um gráfico de linhas, o dado faltante terá um valor diferente de 0
- **Modelos e Estatísticas distorcidos**, como média móvel

Séries Temporais Completas

Solução: inserir as datas faltantes

- Criação de uma relação com todas as datas possíveis (Calendário)
- Calendário LEFT JOIN Relação de Análise
- Dados faltantes terão valores NULL e podem ser tratados

Séries Temporais Completas

Criação do Calendário

- Função GENERATE_SERIES

1

`GENERATE_SERIES(Inicio, Fim, Passos)`

Gera uma relação com dados que vão de um valor inicial até um valor final, com intervalos pré-estabelecidos

- `GENERATE_SERIES(1, 10, 1)` gera a série 1, 2, 3, ..., 10

Séries Temporais Completas

Criação do Calendário

- Função GENERATE_SERIES

```
1  SELECT generate_series(  
2      '2025-04-15'::date,  
3      '2025-04-19'::date,  
4      '1 day'::interval  
5  ) AS dia;
```

dia
2025-04-15 00:00:00
2025-04-16 00:00:00
2025-04-17 00:00:00
2025-04-18 00:00:00
2025-04-19 00:00:00

Séries Temporais Completas - Site

Suponha que a relação dos acessos ao site não possua o dia 25 e que queremos saber o nº de acessos por dia

id_acesso	id_usuario	data_acesso			
1	123	2025-03-20 09:05:11	16	456	2025-03-23 16:40:10
2	456	2025-03-20 09:15:21	17	101	2025-03-23 20:55:00
3	789	2025-03-20 10:30:00	18	123	2025-03-24 09:00:00
4	123	2025-03-20 11:12:45	19	456	2025-03-24 09:10:00
5	101	2025-03-20 14:00:19	20	789	2025-03-24 10:15:20
6	456	2025-03-20 15:22:30	21	222	2025-03-24 11:30:00
7	123	2025-03-20 20:05:00	22	101	2025-03-24 14:20:50
8	456	2025-03-21 08:30:00	23	123	2025-03-24 15:00:00
9	789	2025-03-21 10:45:10	24	456	2025-03-24 16:45:10
10	222	2025-03-21 11:00:00	25	789	2025-03-25 08:55:00
11	123	2025-03-21 13:20:15	26	123	2025-03-25 09:30:00
12	456	2025-03-21 17:50:00	27	456	2025-03-25 10:00:00
13	789	2025-03-22 11:05:00	28	333	2025-03-25 12:10:15
14	333	2025-03-22 15:25:30	29	101	2025-03-25 14:40:00
15	123	2025-03-22 18:00:00	30	222	2025-03-25 15:30:45

Séries Temporais Completas - Site

Suponha que a relação dos acessos ao site não possua o dia 25 e que queremos saber o nº de acessos por dia

```
1  SELECT
2      c.dia_completo,
3      COALESCE(COUNT(*), 0) AS acessos_diarios
4  FROM (
5      SELECT generate_series(
6          '2025-03-20'::date,
7          '2025-03-25'::date,
8          '1 day'::interval
9      )::date AS dia_completo
10 ) c
11 LEFT JOIN acessos_site a ON DATE_TRUNC('day', a.data_acesso) = c.dia_completo
12 GROUP BY c.dia_completo
13 ORDER BY c.dia_completo;
```

Pelo uso do COALESCE, o nº de acessos ao site no dia faltante será substituído por 0

▼ Séries Temporais Completas - Site

dia_completo	acessos_diarios
2025-03-20 00:00:00	7
2025-03-21 00:00:00	5
2025-03-22 00:00:00	3
2025-03-23 00:00:00	2
2025-03-24 00:00:00	7
2025-03-25 00:00:00	0

Janelas Móveis (Rolling Time Windows)

► Janelas Móveis (Rolling Time Windows)

- Usadas em séries temporais
- Permitem suavizar variações de curto prazo
- Identificação de tendências ocultas
- Baseadas em funções de janela (**Window Functions**)
- Aplicações: finanças, acessos, métricas de desempenho



Média Móvel (Moving Average)

- Calcula a média dos últimos **n** períodos
- Suaviza oscilações momentâneas
- Pergunta típica: “Qual a média de receita dos últimos 3 meses?”
- Exemplo: Suponha a tabela VendasMensais:

mes	receita
2025-01-01	10000
2025-02-01	12000
2025-03-01	15000
2025-04-01	13000
2025-05-01	17000

SQL – Média Móvel

- Uso da função **AVG()** com **OVER**
- Sintaxe: **ROWS BETWEEN n PRECEDING AND CURRENT ROW**
- Exemplo: cálculo da média móvel de 3 meses

```
SELECT  
    mes,  
    receita,  
    ROUND(  
        AVG(receita) OVER (  
            ORDER BY mes  
            ROWS BETWEEN 2 PRECEDING AND CURRENT ROW  
        ), 2  
    ) AS media_movel_3m  
FROM VendasMensais;
```

Resultado - Média Móvel

- Receita comparada à média móvel de 3 meses
- Demonstra suavização das variações

mes	receita	media_movel_3m
2025-01-01	10000	10000.00
2025-02-01	12000	11000.00
2025-03-01	15000	12333.33
2025-04-01	13000	13333.33
2025-05-01	17000	15000.00

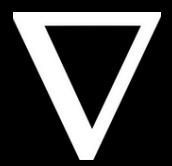
Soma e Contagem Móveis

- Além da média, podemos calcular:
- **Soma móvel**
- **Contagem móvel**
- Exemplo: soma acumulada dos últimos 3 meses

```
SELECT  
    mes,  
    receita,  
    SUM(receita) OVER (  
        ORDER BY mes  
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW  
    ) AS soma_movel_3m  
FROM VendasMensais;
```

Resultado

mes	receita	soma_movel_3m
2025-01-01	10000	10000
2025-02-01	12000	22000
2025-03-01	15000	37000
2025-04-01	13000	40000
2025-05-01	17000	45000



Tipos de Janelas

- **Janela deslizante** (Rolling Window):
últimos **n** registros (ex.: 3 meses)
- **Janela acumulada** (Cumulative Window):
do início até a linha atual
- Exemplo de soma acumulada:

```
SELECT  
    mes,  
    receita,  
    SUM(receita) OVER (ORDER BY mes) AS soma_acumulada  
FROM VendasMensais;
```



Resumo

- Janelas móveis suavizam flutuações
- Permitem identificar tendências
- Funções principais: **AVG, SUM, COUNT**
- Sintaxe clara com **OVER**
- Código mais limpo e eficiente



Comparação com períodos anteriores



Importância da comparação anterior

- Fazer análises temporais
- Não basta valor absoluto → precisamos entender a evolução
- Perguntas respondidas:
 - Receita cresceu ou caiu vs. mês passado?
 - Ano atual vs. ano anterior?
 - Existe sazonalidade?



Funções LAG e LEAD

- LAG: acessa o valor da linha anterior dentro da partição/ordenação definida.
- LEAD: acessa o valor da linha seguinte

Exemplo

mes	receita
2025-01-01	10000
2025-02-01	12000
2025-03-01	15000
2025-04-01	13000

```
SELECT  
    mes,  
    receita,  
    LAG(receita) OVER (ORDER BY mes) AS receita_anterior  
FROM VendasMensais;
```

Resultado:

mes	receita	receita_anterior
2025-01-01	10000	NULL
2025-02-01	12000	10000
2025-03-01	15000	12000
2025-04-01	13000	15000

↓

Crescimento Percentual Período a Período

- Usando LAG para calcular variação %

$$\frac{(receita - receita_anterior)}{receita_anterior} \times 100$$

```
SELECT  
    mes,  
    receita,  
    LAG(receita) OVER (ORDER BY mes) AS receita_anterior,  
    ROUND(  
        (receita - LAG(receita) OVER (ORDER BY mes))  
        / LAG(receita) OVER (ORDER BY mes) * 100, 2  
    ) AS crescimento_percentual  
FROM VendasMensais;
```

Resultado esperado

mes	receita	receita_anterior	crescimento%
2025-01-01	10000	NULL	NULL
2025-02-01	12000	10000	20.00
2025-03-01	15000	12000	25.00
2025-04-01	13000	15000	-13.33

Comparação Anual (Sazonalidade)

- Analisa mesmo mês em anos diferentes
- Detecta sazonalidade
- Exemplo: tabela com 2 anos de dados

mes	receita
2024-01-01	9000
2024-02-01	11000
2025-01-01	10000
2025-02-01	12000

Exemplo

- Comparar **meses iguais em anos diferentes**

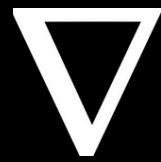
```
SELECT  
    EXTRACT(YEAR FROM mes) AS ano,  
    EXTRACT(MONTH FROM mes) AS mes_num,  
    receita,  
    LAG(receita) OVER (PARTITION BY EXTRACT(MONTH FROM mes)  
                        ORDER BY mes) AS receita_ano_anterior  
FROM ReceitaMensal;
```

- Resultado esperado

ano	mes	receita	receita_ano_anterior
2024	1	9000	NULL
2025	1	10000	9000
2024	2	11000	NULL
2025	2	12000	11000



Análise de Contribuição e Proporção



Análise de Contribuição e Proporção

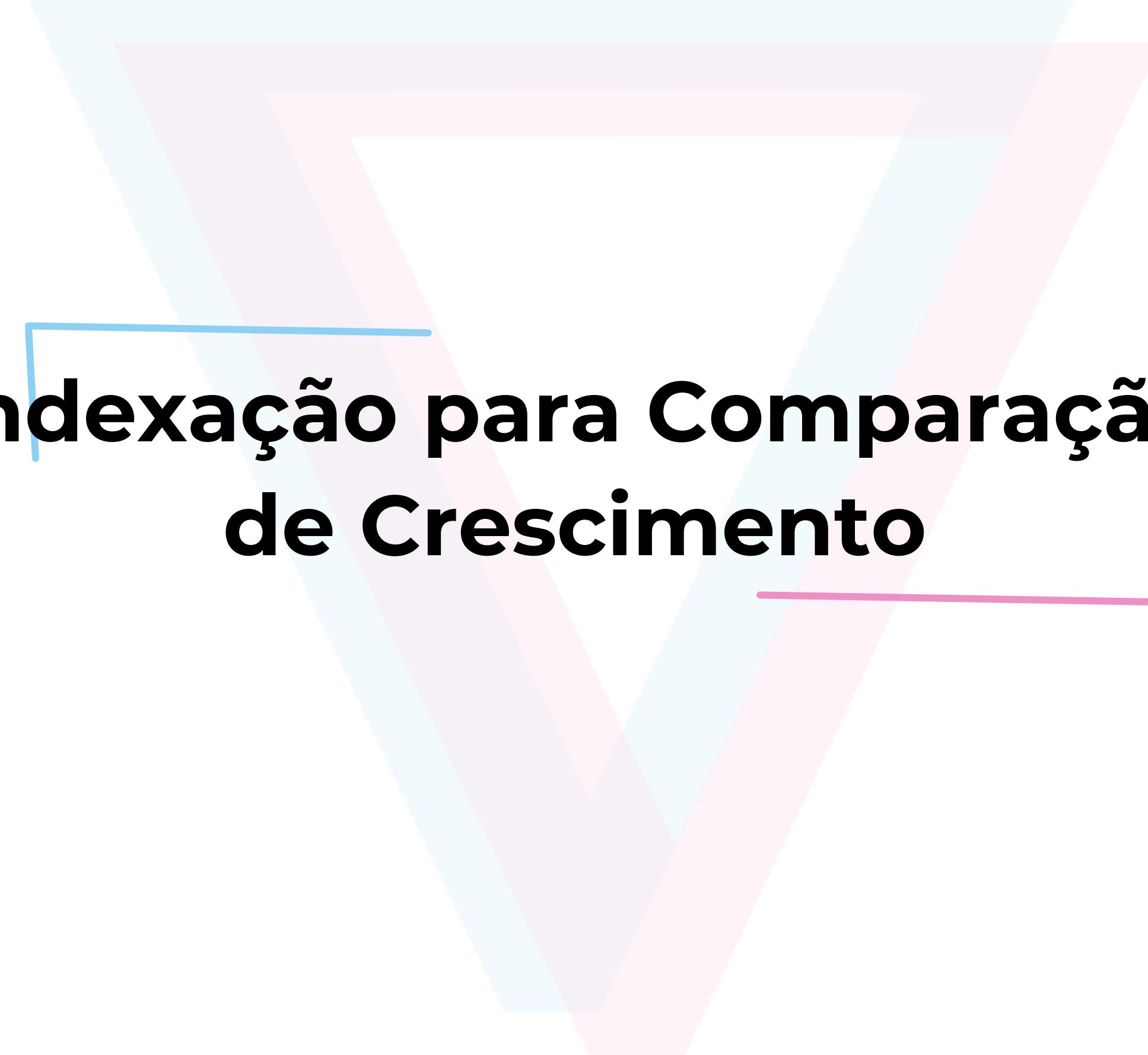
- Analisar a fatia que um certo item contribui para um total. (Por exemplo, quanto um certo produto contribui para a receita.)
- Usar partições para verificar essa fatia em determinado período. (Por exemplo, a fatia do produto em cada mês).

Análise de Contribuição e Proporção

mes	categoria	receita_categoria
2025-01-01	Eletrônicos	5000
2025-01-01		3000
2025-01-01		2000
2025-02-01	Eletrônicos	6000
2025-02-01		2500
2025-02-01		3500

```
SELECT
    mes,
    categoria,
    receita_categoria,
    ROUND(
        (receita_categoria * 100.0 / SUM(receita_categoria) OVER (PARTITION BY mes))
        , 2) AS contribuicao_percentual
FROM
    VendasPorCategoria
ORDER BY
    mes, categoria;
```

mes	categoria	receita_categoria	contribuicao_percentual
2025-01-01	Eletrônicos	5000	50
2025-01-01		3000	30
2025-01-01		2000	20
2025-02-01	Eletrônicos	6000	50
2025-02-01		2500	21
2025-02-01		3500	29



Indexação para Comparação de Crescimento



Comparação de Crescimento

- Precisamos comparar o crescimento de uma série temporal
- Comparar números absolutos é enganoso, é necessário comparar crescimento percentual.

Comparação de Crescimento

mes	id_produto	receita
2025-01-01	A	1000
2025-02-01	A	1200
2025-03-01	A	1500
2025-02-01	B	500
2025-03-01	B	750
2025-04-01	B	1100

Comparação de Crescimento

```
WITH VendasComValorInicial AS (
    SELECT
        mes,
        id_produto,
        receita,
        FIRST_VALUE(receita) OVER (PARTITION BY id_produto ORDER BY mes) AS receita_inicial
    FROM
        VendasDeProdutos
)
SELECT
    mes,
    id_produto,
    receita,
    ROUND(
        (receita * 100.0 / receita_inicial)
    , 2) AS indice_crescimento
FROM
    VendasComValorInicial
ORDER BY
    id_produto, mes;
```

Comparação de Crescimento

mes	categoria	receita_categoria	indice_crescimento
2025-01-01	A	1000	100.00
2025-02-01	A	1200	120.00
2025-03-01	A	1500	150.00
2025-02-01	B	500	100.00
2025-03-01	B	750	150.00
2025-04-01	B	1100	220.00



data@icmc.usp.br



[@data.icmc](https://www.instagram.com/@data.icmc)



[/c/DataICMC](https://www.youtube.com/c/DataICMC)



[/icmc-data](https://github.com/icmc-data)



data.icmc.usp.br

|| obrigado!