

# Relatório Trabalho de Projeto de Banco de Dados

## Integrantes:

- Jorge Coutinho dos Santos Neto
- Rafaela Abrahão de Sá
- Higor Marques de Abreu Souza
- Gabriel Villa Verde Reis

Github: [Projeto de Banco de Dados: E-commerce de jogos](#)

Draw.io: [Modelo Conceitual](#)

## Análise do Banco de Dados

Este documento detalha a estrutura do banco de dados, incluindo uma lista completa de tabelas, seus atributos e os relacionamentos entre elas.

## Tabelas e Atributos

Aqui está a lista de todas as tabelas do banco de dados e os atributos contidos em cada uma.

### 1. usuarios

Armazena informações sobre os usuários da plataforma.

- `id_usuario` (PK): Identificador único do usuário.
- `nome_usuario`: Nome de usuário único.
- `email`: Endereço de e-mail único.
- `senha_hash`: Senha do usuário (armazenada de forma segura).
- `data_registro`: Data e hora do registro.
- `ultimo_login`: Data e hora do último login.

### 2. desenvolvedoras

Armazena informações sobre as empresas desenvolvedoras de jogos.

- `id_desenvolvedora` (PK): Identificador único da desenvolvedora.
- `nome_desenvolvedora`: Nome único da desenvolvedora.
- `site`: Website da desenvolvedora.

### 3. editoras

Armazena informações sobre as empresas editoras de jogos.

- `id_editora` (PK): Identificador único da editora.
- `nome_editora`: Nome único da editora.
- `site`: Website da editora.

## 4. jogos

Tabela central que armazena as informações dos jogos.

- `id_jogo` (PK): Identificador único do jogo.
- `titulo`: Título do jogo.
- `descricao`: Descrição detalhada do jogo.
- `preco`: Preço do jogo.
- `data_lancamento`: Data de lançamento do jogo.
- `id_desenvolvedora` (FK): Chave estrangeira para a tabela `desenvolvedoras`.
- `id_editora` (FK): Chave estrangeira para a tabela `editoras`.
- `url_imagem_capa`: URL da imagem de capa do jogo.
- `classificacao_media`: Média das avaliações dos usuários.

## 5. dlcs

Armazena informações sobre os conteúdos adicionais (DLCs) dos jogos.

- `id_dlc` (PK): Identificador único da DLC.
- `id_jogo_base` (FK): Chave estrangeira para o jogo base ao qual a DLC pertence.
- `titulo_dlc`: Título da DLC.
- `descricao_dlc`: Descrição da DLC.
- `preco_dlc`: Preço da DLC.
- `data_lancamento_dlc`: Data de lançamento da DLC.

## 6. categorias

Armazena as categorias ou gêneros dos jogos.

- `id_categoria` (PK): Identificador único da categoria.
- `nome_categoria`: Nome único da categoria (ex: Ação, RPG, Estratégia).

## 7. promocoões

Armazena informações sobre as promoções aplicadas aos jogos.

- `id_promocao` (PK): Identificador único da promoção.
- `nome_promocao`: Nome da promoção.
- `descricao`: Descrição da promoção.

- `data_inicio`: Data e hora de início da promoção.
- `data_fim`: Data e hora de término da promoção.
- `tipo_desconto`: Tipo de desconto (ex: percentual, valor fixo).
- `valor_desconto`: Valor do desconto.

## 8. noticias

Armazena notícias relacionadas aos jogos.

- `id_noticia` (PK): Identificador único da notícia.
- `titulo`: Título da notícia.
- `conteudo`: Conteúdo da notícia.
- `data_publicacao`: Data e hora da publicação.
- `id_jogo_relacionado` (FK): Chave estrangeira para o jogo ao qual a notícia se refere.

## 9. compras

Armazena o histórico de compras dos usuários.

- `id_compra` (PK): Identificador único da compra.
- `id_usuario` (FK): Chave estrangeira para o usuário que realizou a compra.
- `data_compra`: Data e hora da compra.
- `valor_total`: Valor total da compra.
- `status_compra`: Status da compra (ex: concluída, pendente, cancelada).

## 10. itens\_compra

Tabela associativa que detalha os itens de cada compra.

- `id_item_compra` (PK): Identificador único do item da compra.
- `id_compra` (FK): Chave estrangeira para a compra.
- `id_jogo` (FK): Chave estrangeira para o jogo comprado.
- `id_dlc` (FK): Chave estrangeira para a DLC comprada.
- `preco_unitario`: Preço do item no momento da compra.

## 11. carrinhos\_compra

Armazena o carrinho de compras de cada usuário.

- `id_carrinho` (PK): Identificador único do carrinho.
- `id_usuario` (FK): Chave estrangeira para o usuário dono do carrinho.
- `data_criacao`: Data e hora de criação do carrinho.
- `ultima_atualizacao`: Data e hora da última modificação no carrinho.

## 12. itens\_carrinho

Tabela associativa que detalha os itens no carrinho de compras.

- `id_item_carrinho` (PK): Identificador único do item no carrinho.
- `id_carrinho` (FK): Chave estrangeira para o carrinho de compras.
- `id_jogo` (FK): Chave estrangeira para o jogo no carrinho.
- `id_dlc` (FK): Chave estrangeira para a DLC no carrinho.
- `quantidade`: Quantidade do item no carrinho.
- `preco_no_carrinho`: Preço do item quando foi adicionado ao carrinho.

## 13. jogos\_usuario

Tabela associativa que representa a biblioteca de jogos de cada usuário.

- `id_usuario` (PK, FK): Chave estrangeira para o usuário.
- `id_jogo` (PK, FK): Chave estrangeira para o jogo.
- `data_aquisicao`: Data e hora em que o jogo foi adquirido.

## 14. avaliacoes\_jogo

Armazena as avaliações que os usuários fazem dos jogos que possuem.

- `id_avaliacao` (PK): Identificador único da avaliação.
- `id_usuario` (FK): Chave estrangeira para o usuário que fez a avaliação.
- `id_jogo` (FK): Chave estrangeira para o jogo avaliado.
- `nota`: Nota da avaliação (0 a 10).
- `comentario`: Comentário opcional.
- `data_avaliacao`: Data e hora da avaliação.

## 15. jogo\_categoria

Tabela associativa para o relacionamento N:M entre `jogos` e `categorias`.

- `id_jogo` (PK, FK): Chave estrangeira para o jogo.
- `id_categoria` (PK, FK): Chave estrangeira para a categoria.

## 16. jogo\_promocao

Tabela associativa para o relacionamento N:M entre `jogos` e `promocoes`.

- `id_jogo` (PK, FK): Chave estrangeira para o jogo.
- `id_promocao` (PK, FK): Chave estrangeira para a promoção.

## 17. requisitos\_sistema

Armazena os requisitos de sistema para rodar um jogo.

- `id_requisito` (PK): Identificador único do requisito.
- `id_jogo` (FK): Chave estrangeira para o jogo.
- `os`: Sistema Operacional.
- `processador`: Processador mínimo.
- `memoria_ram`: Memória RAM mínima.
- `placa_video`: Placa de vídeo mínima.
- `armazenamento`: Espaço em disco necessário.
- `notas_adicionais`: Outras informações.

## 18. tickets\_suporte

Armazena os tickets de suporte abertos pelos usuários.

- `id_ticket` (PK): Identificador único do ticket.
- `id_usuario` (FK): Chave estrangeira para o usuário que abriu o ticket.
- `assunto`: Assunto do ticket.
- `descricao`: Descrição do problema.
- `status`: Status do ticket (ex: aberto, em andamento, fechado).
- `data_abertura`: Data e hora de abertura.
- `data_ultima_atualizacao`: Data e hora da última atualização.
- `prioridade`: Prioridade do ticket.

## 19. mensagens\_ticket

Armazena as mensagens trocadas dentro de um ticket de suporte.

- `id_mensagem` (PK): Identificador único da mensagem.
- `id_ticket` (FK): Chave estrangeira para o ticket.
- `id_remetente` (FK): Chave estrangeira para o usuário que enviou a mensagem.
- `conteudo`: Conteúdo da mensagem.
- `data_envio`: Data e hora do envio.

# Relacionamentos e Cardinalidade

A seguir, a lista de todos os relacionamentos entre as tabelas, com suas respectivas cardinalidades.

1. **desenvolvedoras e jogos**
  - **Descrição:** Uma desenvolvedora pode criar vários jogos, mas um jogo é criado por apenas uma desenvolvedora.
  - **Cardinalidade:** `desenvolvedoras (1,1) --- (0,N) jogos`
2. **editoras e jogos**

- **Descrição:** Uma editora pode publicar vários jogos, mas um jogo é publicado por apenas uma editora.
- **Cardinalidade:** editoras (1,1) --- (0,N) jogos
- 3. **jogos e dlcs**
  - **Descrição:** Um jogo pode ter vários DLCs, mas um DLC pertence a um único jogo base.
  - **Cardinalidade:** jogos (1,1) --- (0,N) dlcs
- 4. **jogos e jogo\_categoria**
  - **Descrição:** Um jogo pode estar associado a várias entradas na tabela `jogo_categoria`, uma para cada categoria a que pertence.
  - **Cardinalidade:** jogos (1,1) --- (1,N) jogo\_categoria
- 5. **categorias e jogo\_categoria**
  - **Descrição:** Uma categoria pode estar associada a várias entradas na tabela `jogo_categoria`, uma para cada jogo que pertence a ela.
  - **Cardinalidade:** categorias (1,1) --- (0,N) jogo\_categoria
- 6. **usuarios e compras**
  - **Descrição:** Um usuário pode realizar várias compras, mas uma compra é feita por um único usuário.
  - **Cardinalidade:** usuarios (1,1) --- (0,N) compras
- 7. **compras e itens\_compra**
  - **Descrição:** Uma compra deve conter pelo menos um item, e pode conter vários. Um item de compra pertence a uma única compra.
  - **Cardinalidade:** compras (1,1) --- (1,N) itens\_compra
- 8. **itens\_compra e jogos/dlcs**
  - **Descrição:** Um item de compra pode ser um jogo ou um DLC. Um jogo ou DLC pode estar em vários itens de compra.
  - **Cardinalidade:** itens\_compra (1,1) --- (0,N) jogos (Relacionamento opcional)
  - **Cardinalidade:** itens\_compra (1,1) --- (0,N) dlcs (Relacionamento opcional)
- 9. **usuarios e jogos\_usuario (Biblioteca)**
  - **Descrição:** Um usuário pode ter vários jogos em sua biblioteca, representados por entradas na tabela `jogos_usuario`.
  - **Cardinalidade:** usuarios (1,1) --- (0,N) jogos\_usuario
- 10. **jogos e jogos\_usuario (Biblioteca)**
  - **Descrição:** Um jogo pode pertencer à biblioteca de vários usuários, com cada posse representada por uma entrada em `jogos_usuario`.
  - **Cardinalidade:** jogos (1,1) --- (0,N) jogos\_usuario
- 11. **jogos\_usuario e avaliacoes\_jogo**
  - **Descrição:** Um usuário pode avaliar um jogo que possui. Uma avaliação é feita por um único usuário para um único jogo que ele possui. Um usuário pode fazer no máximo uma avaliação por jogo.
  - **Cardinalidade:** jogos\_usuario (1,1) --- (0,1) avaliacoes\_jogo
- 12. **jogos e requisitos\_sistema**
  - **Descrição:** Um jogo tem um único conjunto de requisitos de sistema.

- **Cardinalidade:** jogos (1,1) --- (1,1) requisitos\_sistema
- 13. **jogos e jogo\_promocao**
  - **Descrição:** Um jogo pode estar associado a várias promoções através da tabela jogo\_promocao.
  - **Cardinalidade:** jogos (1,1) --- (0,N) jogo\_promocao
- 14. **promocoes e jogo\_promocao**
  - **Descrição:** Uma promoção pode incluir vários jogos, cada um representado por uma entrada em jogo\_promocao.
  - **Cardinalidade:** promocoes (1,1) --- (1,N) jogo\_promocao
- 15. **usuarios e carrinhos\_compra**
  - **Descrição:** Um usuário tem um único carrinho de compras.
  - **Cardinalidade:** usuarios (1,1) --- (1,1) carrinhos\_compra
- 16. **carrinhos\_compra e itens\_carrinho**
  - **Descrição:** Um carrinho de compras pode ter vários itens ou nenhum. Um item de carrinho pertence a um único carrinho.
  - **Cardinalidade:** carrinhos\_compra (1,1) --- (0,N) itens\_carrinho
- 17. **itens\_carrinho e jogos/dlcs**
  - **Descrição:** Um item de carrinho pode ser um jogo ou um DLC. Um jogo ou DLC pode estar em vários itens de carrinho.
  - **Cardinalidade:** itens\_carrinho (1,1) --- (0,N) jogos (Relacionamento opcional)
  - **Cardinalidade:** itens\_carrinho (1,1) --- (0,N) dlcs (Relacionamento opcional)
- 18. **jogos e noticias**
  - **Descrição:** Uma notícia pode ser sobre um jogo específico, ou pode não estar relacionada a nenhum jogo. Um jogo pode ter várias notícias associadas.
  - **Cardinalidade:** jogos (1,1) --- (0,N) noticias
- 19. **usuarios e tickets\_suporte**
  - **Descrição:** Um usuário pode abrir vários tickets de suporte, mas um ticket é aberto por um único usuário.
  - **Cardinalidade:** usuarios (1,1) --- (0,N) tickets\_suporte
- 20. **tickets\_suporte e mensagens\_ticket**
  - **Descrição:** Um ticket de suporte deve ter pelo menos uma mensagem (a de abertura) e pode ter várias. Uma mensagem pertence a um único ticket.
  - **Cardinalidade:** tickets\_suporte (1,1) --- (1,N) mensagens\_ticket
- 21. **mensagens\_ticket e usuarios (Remetente)**
  - **Descrição:** Uma mensagem é enviada por um único remetente (usuário ou suporte). Um usuário pode enviar várias mensagens.
  - **Cardinalidade:** usuarios (1,1) --- (0,N) mensagens\_ticket

# Relatório comparativo de tempo de execução das consultas

## Consulta 1:

```
EXPLAIN ANALYSE
SELECT
    u.nome_usuario,
    j.titulo AS titulo_do_jogo,
    c.nome_categoria,
    aj.nota AS nota_atribuida,
    aj.data_avaliacao
FROM
    usuarios u
    JOIN avaliacoes_jogo aj
        ON u.id_usuario = aj.id_usuario
    JOIN jogos j
        ON aj.id_jogo = j.id_jogo
    JOIN jogo_categoria jc
        ON j.id_jogo = jc.id_jogo
    JOIN categorias c
        ON jc.id_categoria = c.id_categoria
WHERE
    u.email = 'usuario_1@emailaleatorio.com'
    AND c.nome_categoria = 'RPG'
    AND aj.nota >= 8
ORDER BY
    aj.data_avaliacao DESC;
```




- **Sem index:**

	QUERY PLAN	
	text	🔒
1	Sort (cost=12029.25..12029.25 rows=1 width=169) (actual time=390.425..416.820 rows=0 loops=1)	
2	Sort Key: aj.data_avaliacao DESC	
3	Sort Method: quicksort Memory: 25kB	
4	-> Nested Loop (cost=1009.18..12029.24 rows=1 width=169) (actual time=390.419..416.814 rows=0 loops=1)	
5	-> Nested Loop (cost=1009.03..12029.05 rows=1 width=55) (actual time=390.418..416.813 rows=0 loops=1)	
6	-> Nested Loop (cost=1008.74..12028.73 rows=1 width=59) (actual time=390.418..416.813 rows=0 loops=1)	
7	-> Gather (cost=1008.45..12028.39 rows=1 width=30) (actual time=390.417..416.811 rows=0 loops=1)	
8	Workers Planned: 2	
9	Workers Launched: 2	
10	-> Hash Join (cost=8.45..11028.29 rows=1 width=30) (actual time=69.976..69.979 rows=0 loops=3)	
11	Hash Cond: (aj.id_usuario = u.id_usuario)	
12	-> Parallel Seq Scan on avaliacoes_jogo aj (cost=0.00..10799.35 rows=83995 width=20) (actual time=0.016..64....	
13	Filter: (nota >= 8)	
14	Rows Removed by Filter: 182095	
15	-> Hash (cost=8.44..8.44 rows=1 width=18) (actual time=0.026..0.027 rows=1 loops=1)	
16	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
17	-> Index Scan using usuarios_email_key on usuarios u (cost=0.42..8.44 rows=1 width=18) (actual time=0.02...	
18	Index Cond: ((email)::text = 'usuario_1@emailaleatorio.com'::text)	
19	-> Index Scan using jogos_pkey on jogos j (cost=0.29..0.33 rows=1 width=29) (never executed)	
20	Index Cond: (id_jogo = aj.id_jogo)	
21	-> Index Only Scan using jogo_categoria_pkey on jogo_categoria jc (cost=0.29..0.31 rows=1 width=8) (never executed)	
22	Index Cond: (id_jogo = aj.id_jogo)	
23	Heap Fetches: 0	
24	-> Index Scan using categorias_pkey on categorias c (cost=0.15..0.17 rows=1 width=122) (never executed)	
25	Index Cond: (id_categoria = jc.id_categoria)	
26	Filter: ((nome_categoria)::text = 'RPG'::text)	
27	Planning Time: 0.723 ms	
28	Execution Time: 416.874 ms	

**Tempo de execução: 416.874ms**

## ● Com index:

	QUERY PLAN	
	text	
1	Sort (cost=33.42..33.43 rows=1 width=169) (actual time=0.038..0.038 rows=0 loops=1)	
2	Sort Key: aj.data_avaliacao DESC	
3	Sort Method: quicksort Memory: 25kB	
4	-> Nested Loop (cost=5.60..33.41 rows=1 width=169) (actual time=0.034..0.035 rows=0 loops=1)	
5	-> Nested Loop (cost=5.46..33.24 rows=1 width=55) (actual time=0.034..0.035 rows=0 loops=1)	
6	-> Nested Loop (cost=5.17..32.92 rows=1 width=59) (actual time=0.034..0.034 rows=0 loops=1)	
7	-> Nested Loop (cost=4.88..32.58 rows=1 width=30) (actual time=0.034..0.034 rows=0 loops=1)	
8	-> Index Scan using idx_usuarios_email on usuarios u (cost=0.42..8.44 rows=1 width=18) (actual time=0.018..0.019 rows=1 loops=1)	
9	Index Cond: ((email)::text = 'usuario_1@emailaleatorio.com')::text	
10	-> Bitmap Heap Scan on avaliacoes_jogo aj (cost=4.46..24.13 rows=1 width=20) (actual time=0.014..0.014 rows=0 loops=1)	
11	Recheck Cond: (u.id_usuario = id_usuario)	
12	Filter: (nota >= 8)	
13	Rows Removed by Filter: 1	
14	Heap Blocks: exact=1	
15	-> Bitmap Index Scan on idx_avaliacoes_jogo_id_usuario (cost=0.00..4.46 rows=5 width=0) (actual time=0.003..0.003 rows=1 loops=1)	
16	Index Cond: (id_usuario = u.id_usuario)	
17	-> Index Scan using jogos_pkey on jogos j (cost=0.29..0.33 rows=1 width=29) (never executed)	
18	Index Cond: (id_jogo = aj.id_jogo)	
19	-> Index Only Scan using jogo_categoria_pkey on jogo_categoria jc (cost=0.29..0.31 rows=1 width=8) (never executed)	
20	Index Cond: (id_jogo = aj.id_jogo)	
21	Heap Fetches: 0	
22	-> Index Scan using categorias_pkey on categorias c (cost=0.14..0.16 rows=1 width=122) (never executed)	
23	Index Cond: (id_categoria = jc.id_categoria)	
24	Filter: ((nome_categoria)::text = 'RPG')::text	
25	Planning Time: 0.758 ms	
26	Execution Time: 0.069 ms	

Tempo de execução: **0.069ms**

## Consulta 2:

EXPLAIN ANALYZE

SELECT

```
e.nome_editora,
COUNT(DISTINCT j.id_jogo) AS total_jogos_publicados,
AVG(av.nota) AS media_geral_avaliacoes,
COUNT(ju.id_jogo) AS total_copias_na_plataforma
```

FROM

```
editoras e
JOIN jogos j
    ON e.id_editora = j.id_editora
LEFT JOIN avaliacoes_jogo av
    ON j.id_jogo = av.id_jogo
LEFT JOIN jogos_usuario ju
    ON j.id_jogo = ju.id_jogo
```

GROUP BY

```
e.id_editora, e.nome_editora
```

ORDER BY

```
total_copias_na_plataforma DESC;
```

- Sem index:

	QUERY PLAN text	
2	Sort Key: (count(ju.id_jogo)) DESC	
3	Sort Method: quicksort Memory: 25kB	
4	-> GroupAggregate (cost=8642422.98..9766943.30 rows=1040 width=70) (actual time=40095.760..61004.443 rows=6 loops=1)	
5	Group Key: e.id_editora	
6	-> Merge Join (cost=8642422.98..9317127.38 rows=44980292 width=34) (actual time=35691.737..46888.799 rows=457266...	
7	Merge Cond: (j.id_editora = e.id_editora)	
8	-> Sort (cost=8642350.47..8754801.20 rows=44980292 width=16) (actual time=35691.687..40957.947 rows=45726686 l...	
9	Sort Key: j.id_editora	
10	Sort Method: external merge Disk: 1163368kB	
11	-> Hash Right Join (cost=31751.46..618602.45 rows=44980292 width=16) (actual time=1114.645..10043.413 rows=4...	
12	Hash Cond: (ju.id_jogo = j.id_jogo)	
13	-> Seq Scan on jogos_usuario ju (cost=0.00..46209.54 rows=2999554 width=4) (actual time=0.029..350.601 rows...	
14	-> Hash (cost=18717.18..18717.18 rows=749783 width=12) (actual time=1114.483..1114.486 rows=749783 loop...	
15	Buckets: 131072 Batches: 16 Memory Usage: 3022kB	
16	-> Hash Right Join (cost=2371.00..18717.18 rows=749783 width=12) (actual time=526.636..943.565 rows=74...	
17	Hash Cond: (av.id_jogo = j.id_jogo)	
18	-> Seq Scan on avaliacoes_jogo av (cost=0.00..14377.83 rows=749783 width=8) (actual time=0.013..91.60...	
19	-> Hash (cost=1746.00..1746.00 rows=50000 width=8) (actual time=526.564..526.565 rows=50000 loops=1)	
20	Buckets: 65536 Batches: 1 Memory Usage: 2466kB	
21	-> Seq Scan on jogos j (cost=0.00..1746.00 rows=50000 width=8) (actual time=491.966..508.512 rows=...	
22	-> Sort (cost=72.52..75.12 rows=1040 width=22) (actual time=0.028..0.034 rows=6 loops=1)	
23	Sort Key: e.id_editora	
24	Sort Method: quicksort Memory: 25kB	
25	-> Seq Scan on editoras e (cost=0.00..20.40 rows=1040 width=22) (actual time=0.017..0.018 rows=6 loops=1)	
26	Planning Time: 0.404 ms	
27	JIT:	
28	Functions: 31	
29	Options: Inlining true, Optimization true, Expressions true, Deforming true	
30	Timing: Generation 2.815 ms, Inlining 14.730 ms, Optimization 283.475 ms, Emission 193.227 ms, Total 494.247 ms	
31	Execution Time: 61305.019 ms	
Total rows: 31 Query complete 00:01:02.601		

Tempo de execução: 61305ms

- **Com index:**

	QUERY PLAN text	
1	Sort (cost=3561584.89..3561587.49 rows=1040 width=70) (actual time=42831.554..42831.559 rows=6 loops=1)	
2	Sort Key: (count(ju.id_jogo)) DESC	
3	Sort Method: quicksort Memory: 25kB	
4	-> GroupAggregate (cost=5649.42..3561532.77 rows=1040 width=70) (actual time=9716.817..42831.534 rows=6 loops=1)	
5	Group Key: e.id_editora	
6	-> Nested Loop Left Join (cost=5649.42..3111716.85 rows=44980292 width=34) (actual time=329.585..30593.095 rows=45726686 loops=1)	
7	-> Nested Loop Left Join (cost=5648.99..85412.71 rows=749783 width=30) (actual time=329.534..3566.042 rows=749783 loops=1)	
8	-> Merge Join (cost=5648.56..6398.71 rows=50000 width=26) (actual time=329.483..412.338 rows=50000 loops=1)	
9	Merge Cond: (e.id_editora = j.id_editora)	
10	-> Index Scan using editoras_pkey on editoras e (cost=0.15..36.75 rows=1040 width=22) (actual time=0.012..0.021 rows=6 loops=1)	
11	-> Sort (cost=5648.41..5773.41 rows=50000 width=8) (actual time=26.939..61.006 rows=50000 loops=1)	
12	Sort Key: j.id_editora	
13	Sort Method: quicksort Memory: 3710kB	
14	-> Seq Scan on jogos j (cost=0.00..1746.00 rows=50000 width=8) (actual time=0.020..15.549 rows=50000 loops=1)	
15	-> Index Scan using idx_avaliacoes_jogo_id_jogo on avaliacoes_jogo av (cost=0.42..1.42 rows=16 width=8) (actual time=0.009..0.060 rows=15..)	
16	Index Cond: (id_jogo = j.id_jogo)	
17	-> Index Only Scan using idx_jogos_usuario_id_jogo on jogos_usuario ju (cost=0.43..3.44 rows=60 width=4) (actual time=0.004..0.031 rows=61 lo...)	
18	Index Cond: (id_jogo = j.id_jogo)	
19	Heap Fetches: 45726686	
20	Planning Time: 0.675 ms	
21	JIT:	
22	Functions: 20	
23	Options: Inlining true, Optimization true, Expressions true, Deforming true	
24	Timing: Generation 2.114 ms, Inlining 15.979 ms, Optimization 149.299 ms, Emission 136.655 ms, Total 304.046 ms	
25	Execution Time: 42833.825 ms	

**Tempo de execução: 42833ms**

## Consulta 3:

EXPLAIN ANALYZE

```
WITH JogosDoUsuarioAlvo AS (
    SELECT id_jogo
    FROM jogos_usuario
    WHERE id_usuario = 123
),
UsuariosSimilares AS (
    SELECT DISTINCT ju.id_usuario
    FROM jogos_usuario ju
    WHERE ju.id_jogo IN (SELECT id_jogo FROM JogosDoUsuarioAlvo)
    AND ju.id_usuario != 123
)
SELECT
    j.titulo,
    COUNT(ju.id_usuario) AS popularidade_entre_similares
FROM
    jogos_usuario j
    JOIN jogos_usuario ju
        ON ju.id_jogo = j.id_jogo
```

WHERE

```
ju.id_usuario IN (SELECT id_usuario FROM UsuariosSimilares)
AND ju.id_jogo NOT IN (SELECT id_jogo FROM JogosDoUsuarioAlvo)
```

GROUP BY

```
j.id_jogo, j.titulo
```

ORDER BY

```
popularidade_entre_similares DESC
```

LIMIT 15;

- **Sem index:**

	QUERY PLAN	
	text	🔒
1	Limit (cost=111973.32..111973.36 rows=15 width=37) (actual time=724.062..724.071 rows=15 loops=1)	
2	CTE jogosdousuarioalvo	
3	-> Bitmap Heap Scan on jogos_usuario (cost=4.55..67.25 rows=16 width=4) (actual time=0.020..0.033 rows=9 loops=1)	
4	Recheck Cond: (id_usuario = 123)	
5	Heap Blocks: exact=9	
6	-> Bitmap Index Scan on jogos_usuario_pkey (cost=0.00..4.55 rows=16 width=0) (actual time=0.014..0.015 rows=9 loops=1)	
7	Index Cond: (id_usuario = 123)	
8	-> Sort (cost=111906.08..111925.46 rows=7751 width=37) (actual time=710.259..710.264 rows=15 loops=1)	
9	Sort Key: (count(ju.id_usuario)) DESC	
10	Sort Method: top-N heapsort Memory: 26kB	
11	-> GroupAggregate (cost=108803.39..111715.91 rows=7751 width=37) (actual time=688.027..708.654 rows=7477 loops=1)	
12	Group Key: j.id_jogo	
13	-> Merge Join (cost=108803.39..111599.65 rows=7751 width=33) (actual time=688.017..705.835 rows=8138 loops=1)	
14	Merge Cond: (j.id_jogo = ju.id_jogo)	
15	-> Index Scan using jogos_pkey on jogos j (cost=0.29..2555.29 rows=50000 width=29) (actual time=0.018..11.034 rows=...	
16	-> Sort (cost=108803.10..108822.47 rows=7751 width=8) (actual time=687.972..689.018 rows=8138 loops=1)	
17	Sort Key: ju.id_jogo	
18	Sort Method: quicksort Memory: 574kB	
19	-> Nested Loop (cost=61642.03..108302.38 rows=7751 width=8) (actual time=672.410..686.240 rows=8138 loops=1)	
20	-> Unique (cost=61641.24..61646.06 rows=964 width=4) (actual time=668.394..668.624 rows=541 loops=1)	
21	-> Sort (cost=61641.24..61643.65 rows=964 width=4) (actual time=668.392..668.511 rows=543 loops=1)	
22	Sort Key: ju_1.id_usuario	
23	Sort Method: quicksort Memory: 50kB	
24	-> Hash Semi Join (cost=0.52..61593.46 rows=964 width=4) (actual time=0.339..666.652 rows=543 loops=...	
25	Hash Cond: (ju_1.id_jogo = jogosdousuarioalvo.id_jogo)	
26	-> Seq Scan on jogos_usuario ju_1 (cost=0.00..53708.43 rows=2999538 width=8) (actual time=0.055..3...	
27	Filter: (id_usuario <> 123)	
28	Rows Removed by Filter: 9	
29	-> Hash (cost=0.32..0.32 rows=16 width=4) (actual time=0.038..0.039 rows=9 loops=1)	
30	Buckets: 1024 Batches: 1 Memory Usage: 9kB	

30	Buckets: 1024 Batches: 1 Memory Usage: 9kB
31	-> CTE Scan on jogosdousuarioalvo (cost=0.00..0.32 rows=16 width=4) (actual time=0.023..0.036 ro...
32	-> Index Only Scan using jogos_usuario_pkey on jogos_usuario ju (cost=0.79..48.31 rows=8 width=8) (actual time=...
33	Index Cond: (id_usuario = ju_1.id_usuario)
34	Filter: (NOT (hashed SubPlan 2))
35	Rows Removed by Filter: 1
36	Heap Fetches: 8681
37	SubPlan 2
38	-> CTE Scan on jogosdousuarioalvo jogosdousuarioalvo_1 (cost=0.00..0.32 rows=16 width=4) (actual time=0....
39	Planning Time: 0.433 ms
40	JIT:
41	Functions: 42
42	Options: Inlining false, Optimization false, Expressions true, Deforming true
43	Timing: Generation 3.384 ms, Inlining 0.000 ms, Optimization 0.821 ms, Emission 15.908 ms, Total 20.113 ms
44	Execution Time: 727.053 ms
Total rows: 44 Query complete 00:00:00.772	

Tempo de execução: **727.053ms**

- **Com index:**

	QUERY PLAN text	
9	Sort Key: (count(ju.id_usuario)) DESC	
10	Sort Method: top-N heapsort Memory: 26kB	
11	-> HashAggregate (cost=52783.19..52860.70 rows=7751 width=37) (actual time=48.672..50.323 rows=7477 loops=1)	
12	Group Key: j.id_jogo	
13	-> Nested Loop (cost=3711.50..52744.44 rows=7751 width=33) (actual time=1.501..45.208 rows=8138 loops=1)	
14	-> Nested Loop (cost=3711.21..50332.38 rows=7751 width=8) (actual time=1.482..24.589 rows=8138 loops=1)	
15	-> HashAggregate (cost=3710.42..3720.06 rows=964 width=4) (actual time=1.345..1.615 rows=541 loops=1)	
16	Group Key: ju_1.id_usuario	
17	-> Nested Loop (cost=5.25..3708.01 rows=964 width=4) (actual time=0.083..1.181 rows=543 loops=1)	
18	-> HashAggregate (cost=0.36..0.52 rows=16 width=4) (actual time=0.059..0.064 rows=9 loops=1)	
19	Group Key: jogosdousuarioalvo.id_jogo	
20	-> CTE Scan on jogosdousuarioalvo (cost=0.00..0.32 rows=16 width=4) (actual time=0.030..0.055 rows=9 loops=1)	
21	-> Bitmap Heap Scan on jogos_usuario ju_1 (cost=4.89..231.12 rows=60 width=8) (actual time=0.021..0.117 rows=60 loops=9)	
22	Recheck Cond: (id_jogo = jogosdousuarioalvo.id_jogo)	
23	Filter: (id_usuario <> 123)	
24	Rows Removed by Filter: 1	
25	Heap Blocks: exact=550	
26	-> Bitmap Index Scan on idx_jogos_usuario_id_jogo_id_usuario (cost=0.00..4.88 rows=60 width=0) (actual time=0.015..0.015 rows=61 loops=9)	
27	Index Cond: (id_jogo = jogosdousuarioalvo.id_jogo)	
28	-> Index Only Scan using idx_jogos_usuario_id_usuario_id_jogo on jogos_usuario ju (cost=0.79..48.26 rows=8 width=8) (actual time=0.011..0.040 ro...	
29	Index Cond: (id_usuario = ju_1.id_usuario)	
30	Filter: (NOT (hashed SubPlan 2))	
31	Rows Removed by Filter: 1	
32	Heap Fetches: 8681	
33	SubPlan 2	
34	-> CTE Scan on jogosdousuarioalvo jogosdousuarioalvo_1 (cost=0.00..0.32 rows=16 width=4) (actual time=0.001..0.002 rows=9 loops=1)	
35	-> Index Scan using jogos_pkey on jogos j (cost=0.29..0.31 rows=1 width=29) (actual time=0.002..0.002 rows=1 loops=8138)	
36	Index Cond: (id_jogo = ju.id_jogo)	
37	Planning Time: 0.680 ms	
38	Execution Time: 51.668 ms	
Total rows: 38 Query complete 00:00:00.102		

Tempo de execução: **51.668ms**

## Consulta 4:

EXPLAIN ANALYZE

SELECT

u.id\_usuario,

u.email

FROM

usuarios u

JOIN jogos\_usuario ju\_base

ON u.id\_usuario = ju\_base.id\_usuario

AND ju\_base.id\_jogo = 45

WHERE

NOT EXISTS (

SELECT 1

FROM jogos\_usuario ju\_dlc

WHERE ju\_dlc.id\_usuario = u.id\_usuario

AND ju\_dlc.id\_jogo = (SELECT d.id\_dlc  
FROM dlcs d  
WHERE d.id\_dlc = 12)

);

- Sem index:

	QUERY PLAN	
	text	
1	Gather (cost=1009.15..33082.04 rows=60 width=37) (actual time=13.744..207.545 rows=47 loops=1)	
2	Workers Planned: 2	
3	Params Evaluated: \$0	
4	Workers Launched: 2	
5	InitPlan 1 (returns \$0)	
6	-> Index Only Scan using dlcs_pkey on dlcs d (cost=0.29..8.30 rows=1 width=4) (actual time=0.005..0.007 rows=1 loops=1)	
7	Index Cond: (id_dlc = 12)	
8	Heap Fetches: 1	
9	-> Nested Loop Anti Join (cost=0.85..32067.74 rows=25 width=37) (actual time=5.018..171.425 rows=16 loops=3)	
10	-> Nested Loop (cost=0.42..32042.61 rows=25 width=37) (actual time=4.987..169.977 rows=16 loops=3)	
11	-> Parallel Seq Scan on jogos_usuario ju_base (cost=0.00..31836.68 rows=25 width=4) (actual time=4.954..169.696 rows=16 loops=3)	
12	Filter: (id_jogo = 45)	
13	Rows Removed by Filter: 999836	
14	-> Index Scan using usuarios_pkey on usuarios u (cost=0.42..8.24 rows=1 width=37) (actual time=0.011..0.011 rows=1 loops=47)	
15	Index Cond: (id_usuario = ju_base.id_usuario)	
16	-> Index Only Scan using jogos_usuario_pkey on jogos_usuario ju_dlc (cost=0.43..0.98 rows=1 width=4) (actual time=0.090..0.090 rows=0 loops=1)	
17	Index Cond: ((id_usuario = u.id_usuario) AND (id_jogo = \$0))	
18	Heap Fetches: 0	
19	Planning Time: 0.409 ms	
20	Execution Time: 207.600 ms	

Tempo de execução: 207.600ms

- Com index:

	QUERY PLAN	
	text	
1	Nested Loop Anti Join (cost=14.05..794.09 rows=60 width=37) (actual time=0.061..0.344 rows=47 loops=1)	
2	InitPlan 1 (returns \$0)	
3	-> Index Only Scan using dics_pkey on dics d (cost=0.29..8.30 rows=1 width=4) (actual time=0.007..0.008 rows=1 loops=1)	
4	Index Cond: (id_dlc = 12)	
5	Heap Fetches: 1	
6	-> Nested Loop (cost=5.31..728.95 rows=60 width=37) (actual time=0.040..0.244 rows=47 loops=1)	
7	-> Bitmap Heap Scan on jogos_usuario ju_base (cost=4.89..234.70 rows=60 width=4) (actual time=0.031..0.089 rows=47 loops=1)	
8	Recheck Cond: (id_jogo = 45)	
9	Heap Blocks: exact=47	
10	-> Bitmap Index Scan on idx_jogos_usuario_id_jogo_id_usuario (cost=0.00..4.88 rows=60 width=0) (actual time=0.021..0.022 rows=47 loops=1)	
11	Index Cond: (id_jogo = 45)	
12	-> Index Scan using usuarios_pkey on usuarios u (cost=0.42..8.24 rows=1 width=37) (actual time=0.003..0.003 rows=1 loops=47)	
13	Index Cond: (id_usuario = ju_base.id_usuario)	
14	-> Index Only Scan using idx_jogos_usuario_id_jogo_id_usuario on jogos_usuario ju_dlc (cost=0.43..0.94 rows=1 width=4) (actual time=0.002..0.002 rows=0 loops=1)	
15	Index Cond: ((id_jogo = \$0) AND (id_usuario = u.id_usuario))	
16	Heap Fetches: 0	
17	Planning Time: 0.687 ms	
18	Execution Time: 0.381 ms	

Tempo de execução: **0.381ms**

## Consulta 5:

```
EXPLAIN ANALYZE
SELECT titulo, preco, descricao
FROM jogos
WHERE titulo ILIKE '%Sombras%';
```

- Sem index:

	QUERY PLAN	
	text	
1	Seq Scan on jogos (cost=0.00..1871.00 rows=7071 width=83) (actual time=0.016..43.340 rows=6982 loops=1)	
2	Filter: ((titulo)::text ~* '%Sombras%':text)	
3	Rows Removed by Filter: 43018	
4	Planning Time: 0.157 ms	
5	Execution Time: 43.604 ms	

Tempo de execução: **43.604ms**

- Com index:

	QUERY PLAN	
	text	
1	Bitmap Heap Scan on jogos (cost=106.80..1441.19 rows=7071 width=83) (actual time=1.300..8.632 rows=6982 loops=1)	
2	Recheck Cond: ((titulo)::text ~* '%Sombras%':text)	
3	Heap Blocks: exact=1241	
4	-> Bitmap Index Scan on idx_gin_jogos_titulo (cost=0.00..105.03 rows=7071 width=0) (actual time=1.147..1.147 rows=6982 loops=1)	
5	Index Cond: ((titulo)::text ~* '%Sombras%':text)	
6	Planning Time: 0.203 ms	
7	Execution Time: 8.870 ms	

Tempo de execução: **8.870ms**



## Consulta 6:

EXPLAIN ANALYZE

SELECT

```
COUNT(id_compra) AS numero_de_compras,  
SUM(valor_total) AS faturamento_no_periodo
```

FROM

compras

WHERE

```
data_compra >= NOW() - INTERVAL '30 days';
```

- Sem index:

	QUERY PLAN text
1	Aggregate (cost=753.32..753.33 rows=1 width=40) (actual time=7.219..7.221 rows=1 loops=1)
2	-> Seq Scan on compras (cost=0.00..751.00 rows=464 width=10) (actual time=0.010..7.132 rows=461 loop...
3	Filter: (data_compra >= (now() - '30 days'::interval))
4	Rows Removed by Filter: 29539
5	Planning Time: 0.687 ms
6	Execution Time: 7.244 ms

Tempo de execução: 7.244ms

- Com index:

	QUERY PLAN text
1	Aggregate (cost=248.33..248.34 rows=1 width=40) (actual time=0.411..0.412 rows=1 loops=1)
2	-> Bitmap Heap Scan on compras (cost=11.89..246.01 rows=464 width=10) (actual time=0.094..0.347 rows=461 loops=1)
3	Recheck Cond: (data_compra >= (now() - '30 days'::interval))
4	Heap Blocks: exact=198
5	-> Bitmap Index Scan on idx_compras_data_compra (cost=0.00..11.77 rows=464 width=0) (actual time=0.073..0.074 rows=461 l...
6	Index Cond: (data_compra >= (now() - '30 days'::interval))
7	Planning Time: 0.305 ms
8	Execution Time: 0.439 ms

Tempo de execução: 0.439ms

## Consulta 7:

EXPLAIN ANALYZE

SELECT

```
j.titulo,  
(SELECT COUNT(*)  
FROM itens_compra ic  
WHERE ic.id_jogo = j.id_jogo) AS total_de_vendas
```

FROM

```
jogos j
ORDER BY
j.titulo;
```

- **Sem index:**

	QUERY PLAN text	
1	Sort (cost=47207398.41..47207523.41 rows=50000 width=33) (actual time=130809.028..130829.899 rows=50000 loops=1)	
2	Sort Key: j.titulo	
3	Sort Method: external merge Disk: 2336kB	
4	-> Seq Scan on jogos j (cost=0.00..47203496.00 rows=50000 width=33) (actual time=45.306..130467.772 rows=50000 loops=1)	
5	SubPlan 1	
6	-> Aggregate (cost=944.02..944.03 rows=1 width=8) (actual time=2.603..2.603 rows=1 loops=50000)	
7	-> Seq Scan on itens_compra ic (cost=0.00..944.00 rows=10 width=0) (actual time=2.337..2.593 rows=1 loops=50000)	
8	Filter: (id_jogo = j.id_jogo)	
9	Rows Removed by Filter: 49999	
10	Planning Time: 0.122 ms	
11	JIT:	
12	Functions: 8	
13	Options: Inlining true, Optimization true, Expressions true, Deforming true	
14	Timing: Generation 0.741 ms, Inlining 6.325 ms, Optimization 22.200 ms, Emission 14.132 ms, Total 43.398 ms	
15	Execution Time: 130833.621 ms	
Total rows: 15		Query complete 00:02:11.125

**Tempo de execução: 130833.621ms**

- **Com index:**

	QUERY PLAN text	
1	Sort (cost=1966443.07..1966568.07 rows=50000 width=33) (actual time=593.102..614.946 rows=50000 loops=1)	
2	Sort Key: j.titulo	
3	Sort Method: external merge Disk: 2336kB	
4	-> Seq Scan on jogos j (cost=0.00..1962540.66 rows=50000 width=33) (actual time=56.763..289.948 rows=50000 loops=1)	
5	SubPlan 1	
6	-> Aggregate (cost=39.21..39.22 rows=1 width=8) (actual time=0.004..0.004 rows=1 loops=50000)	
7	-> Bitmap Heap Scan on itens_compra ic (cost=4.37..39.18 rows=10 width=0) (actual time=0.001..0.003 rows=1 loops=50000)	
8	Recheck Cond: (id_jogo = j.id_jogo)	
9	Heap Blocks: exact=49228	
10	-> Bitmap Index Scan on idx_itens_compra_id_jogo (cost=0.00..4.37 rows=10 width=0) (actual time=0.001..0.001 rows=1 loops=50000)	
11	Index Cond: (id_jogo = j.id_jogo)	
12	Planning Time: 0.236 ms	
13	JIT:	
14	Functions: 10	
15	Options: Inlining true, Optimization true, Expressions true, Deforming true	
16	Timing: Generation 1.014 ms, Inlining 6.726 ms, Optimization 24.294 ms, Emission 25.450 ms, Total 57.483 ms	
17	Execution Time: 618.365 ms	
Total rows: 17		Query complete 00:00:00.649

**Tempo de execução: 618.365ms**

## Consulta 8:

```
EXPLAIN ANALYZE
SELECT
```

```

COUNT(*)
FROM
  jogos_usuario
WHERE
  id_jogo = 50;

```

- **Sem index:**

	QUERY PLAN	
	text	
1	Finalize Aggregate (cost=32836.95..32836.96 rows=1 width=8) (actual time=134.887..140.106 rows=1 loops=1)	
2	-> Gather (cost=32836.74..32836.95 rows=2 width=8) (actual time=134.773..140.101 rows=3 loops=1)	
3	Workers Planned: 2	
4	Workers Launched: 2	
5	-> Partial Aggregate (cost=31836.74..31836.75 rows=1 width=8) (actual time=118.638..118.639 rows=1 loops=3)	
6	-> Parallel Seq Scan on jogos_usuario (cost=0.00..31836.68 rows=25 width=0) (actual time=1.143..118.585 rows=21 loops=3)	
7	Filter: (id_jogo = 50)	
8	Rows Removed by Filter: 999831	
9	Planning Time: 0.105 ms	
10	Execution Time: 140.133 ms	
Total rows: 10    Query complete 00:00:00.174		

**Tempo de execução: 140.133ms**

- **Com index:**

	QUERY PLAN	
	text	
1	Aggregate (cost=234.85..234.86 rows=1 width=8) (actual time=0.114..0.115 rows=1 loops=1)	
2	-> Bitmap Heap Scan on jogos_usuario (cost=4.89..234.70 rows=60 width=0) (actual time=0.043..0.108 rows=62 loops=1)	
3	Recheck Cond: (id_jogo = 50)	
4	Heap Blocks: exact=62	
5	-> Bitmap Index Scan on idx_jogos_usuario_id_jogo_id_usuario (cost=0.00..4.88 rows=60 width=0) (actual time=0.024..0.024 rows=62 loops=1)	
6	Index Cond: (id_jogo = 50)	
7	Planning Time: 0.187 ms	
8	Execution Time: 0.136 ms	
Total rows: 8    Query complete 00:00:00.074		

**Tempo de execução: 0.136ms**

## Consulta 9:

```

EXPLAIN ANALYZE
SELECT
  id_ticket,
  assunto,
  data_abertura
FROM
  tickets_suporte
WHERE
  status = 'ABERTO'
  AND prioridade = 'ALTA';

```

- **Sem index:**

	QUERY PLAN	
	text	
1	Seq Scan on tickets_suporte (cost=0.00..2264.74 rows=18 width=42) (actual time=0.010..6.584 rows=206 loops=...	
2	Filter: (((status)::text = 'ABERTO'::text) AND ((prioridade)::text = 'ALTA'::text))	
3	Rows Removed by Filter: 51777	
4	Planning Time: 0.093 ms	
5	Execution Time: 6.657 ms	
Total rows: 5		Query complete 00:00:00.066

**Tempo de execução:**6.657ms

- **Com index:**

	QUERY PLAN	
	text	
1	Index Scan using idx_tickets_abertos_prioridade_alta on tickets_suporte (cost=0.14..8.42 rows=18 width=42) (actual time=0.014..0.092 rows=206 loops=1)	
2	Planning Time: 0.263 ms	
3	Execution Time: 0.123 ms	
Total rows: 3		Query complete 00:00:00.075

**Tempo de execução:**0.123ms

## Consulta 10:

```

EXPLAIN ANALYZE
SELECT
    c.id_compra,
    c.id_usuario,
    c.data_compra,
    i.id_jogo
FROM
    compras c
JOIN
    itens_compra i ON c.id_compra = i.id_compra
WHERE
    c.data_compra = '2024-07-10';

```

- **Sem index:**

	QUERY PLAN text	
1	Hash Join (cost=601.01..1551.28 rows=2 width=20) (actual time=2.170..2.172 rows=0 loops=1)	
2	Hash Cond: (i.id_compra = c.id_compra)	
3	-> Seq Scan on itens_compra i (cost=0.00..819.00 rows=50000 width=8) (actual time=0.008..0.009 rows=1 loops=1)	
4	-> Hash (cost=601.00..601.00 rows=1 width=16) (actual time=2.159..2.160 rows=0 loops=1)	
5	Buckets: 1024 Batches: 1 Memory Usage: 8kB	
6	-> Seq Scan on compras c (cost=0.00..601.00 rows=1 width=16) (actual time=2.158..2.158 rows=0 loops=1)	
7	Filter: (data_compra = '2024-07-10 00:00:00':timestamp without time zone)	
8	Rows Removed by Filter: 30000	
9	Planning Time: 0.234 ms	
10	Execution Time: 2.194 ms	
Total rows: 10		Query complete 00:00:00.080

**Tempo de execução: 2.194ms**

- **Com index:**

	QUERY PLAN text	
1	Hash Join (cost=8.32..958.59 rows=2 width=20) (actual time=0.023..0.024 rows=0 loops=1)	
2	Hash Cond: (i.id_compra = c.id_compra)	
3	-> Seq Scan on itens_compra i (cost=0.00..819.00 rows=50000 width=8) (actual time=0.007..0.008 rows=1 loops=1)	
4	-> Hash (cost=8.30..8.30 rows=1 width=16) (actual time=0.013..0.013 rows=0 loops=1)	
5	Buckets: 1024 Batches: 1 Memory Usage: 8kB	
6	-> Index Scan using idx_compras_data_compra on compras c (cost=0.29..8.30 rows=1 width=16) (actual time=0.013..0.013 rows=0 l...	
7	Index Cond: (data_compra = '2024-07-10 00:00:00':timestamp without time zone)	
8	Planning Time: 0.309 ms	
9	Execution Time: 0.043 ms	
Total rows: 9		Query complete 00:00:00.075

**Tempo de execução: 0.043ms**