Dataset fictício para desafio: Análise de Redes Sociais (Neo4j)

Este documento contém:

- CSVs de exemplo (conteúdo pronto para salvar como arquivos `.csv` no diretório `import/` do Neo4j).
- Scripts Cypher para criar nós e relacionamentos usando `LOAD CSV WITH HEADERS`.
- Consultas de exemplo para extrair insights (recomendações, post mais curtido, menor caminho, detecção de comunidades).

Observação: copie cada bloco CSV para um arquivo separado (por exemplo `users.csv`) e coloque-os na pasta `import/` da sua instalação Neo4j (ou use `neo4j-admin import` para importação em massa). Os scripts Cypher consideram que os arquivos estão em `file:///users.csv`, etc.

Arquivos CSV (salve cada bloco como o respectivo arquivo)

`users.csv`

```
user_id,name,age,city,occupation,interests
u1,Ana Silva,29,Jacarezinho,Data Analyst,Python;Graphs;Music
u2,Bruno Costa,35,Piraju,Software Engineer,Neo4j;Databases;Soccer
u3,Carla Pereira,24,São Paulo,Marketing,Design;Influence;Music
u4,Diego Rocha,42,Campinas,Product Manager,Business;Startups
u5,Elisa Gomes,31,Jacarezinho,Social Worker,Policy;Community;Health
u6,Felipe Alves,27,Piraju,Student,Algorithms;Music
u7,Gisele Santos,38,São Paulo,Researcher,Graphs;Science
u8,Hugo Lima,22,Jacarezinho,Intern,Python;Football
u9,Ivana Ribeiro,45,Campinas,HR,People;Recruiting
u10,João Martins,33,Piraju,DevOps,Cloud;Linux
u11,Karen Duarte,28,São Paulo,Designer,UX;Design;Photos
u12,Lucas Nogueira,30,Jacarezinho,Developer,JS;Music
u13,Maria Oliveira,40,Campinas,Teacher,Education;Community
u14,Nando Ferreira,26,São Paulo,Data Scientist,ML;Graphs
u15,Olga Pinto,50,Jacarezinho,Counselor,Health;Community
u16,Paulo Costa,37,Piraju,Entrepreneur,Business;Startups
u17,Queila Ramos,23,São Paulo,Student,Design;Music
u18,Rafael Torres,34,Campinas,Analyst,Finance;Data
u19,Susana Melo,29,Piraju,Community Manager,Events;Groups
u20,Thiago Souza,41,São Paulo,Engineer,Networks;Systems
```

`groups.csv`

```
group_id,name,topic,created_at
g1,Data Graphs,Banco de dados grafo,2024-01-10
g2,Music Lovers,Música e playlists,2023-11-05
g3,Community Support,Ajuda local e voluntariado,2022-08-21
g4,StartupsBR,Empreendedorismo e startups,2023-03-30
g5,Design Corner,Design e UX,2024-05-12
```

`posts.csv`

```
post_id,author_id,content,created_at
p1,u2,"Como otimizar queries no Neo4j: 5 dicas rápidas",2025-09-02
p2,u1,"Visualizando redes com Python e NetworkX",2025-09-10
p3,u3,"Criando uma campanha de marketing de sucesso",2025-10-01
p4,u5,"Como mapear vulnerabilidades em comunidades locais",2025-09-27
p5,u14,"Explicando PageRank em grafos sociais",2025-10-12
p6,u12,"Playlist da semana: indie e eletrônico",2025-10-05
p7,u2,"Benchmark: import CSV vs neo4j-admin import",2025-10-20
p8,u19,"Próximo encontro do grupo Community Support",2025-10-15
p9,u6,"Roteiro de estudos: algoritmos e estruturas de dados",2025-09-18
p10,u11,"Tendências em UX para 2026",2025-10-22
```

`comments.csv`

```
comment_id,post_id,author_id,content,created_at
c1,p1,u14,"Ótimas dicas! Testei a 2 e melhorou bastante.",2025-09-03
c2,p2,u12,"Valeu pelo notebook, muito útil.",2025-09-11
c3,p3,u11,"Boa estratégia de segmentação!",2025-10-02
c4,p4,u13,"Posso ajudar com contatos locais.",2025-09-28
c5,p5,u7,"Explicação clara — e o exemplo numérico ajudou.",2025-10-13
c6,p6,u17,"Adorei a seleção 🔥",2025-10-06
c7,p1,u10,"Concordo, a dica 4 é essencial.",2025-09-04
c8,p8,u5,"Confirmo presença no encontro.",2025-10-16
c9,p9,u6,"Segue meu repositório com exercícios.",2025-09-19
c10,p7,u2,"Adicionei benchs com 1M de nós.",2025-10-21
```

`likes.csv` **(somente likes em posts)**

```
like_id,post_id,user_id,created_at
l1,p1,u1,2025-09-03
l2,p1,u14,2025-09-03
```

```
l3,p1,u10,2025-09-04
l4,p2,u12,2025-09-11
l5,p2,u6,2025-09-11
l6,p3,u11,2025-10-02
l7,p3,u1,2025-10-02
l8,p5,u7,2025-10-13
l9,p5,u14,2025-10-13
l10,p6,u17,2025-10-06
l11,p8,u5,2025-10-16
l12,p8,u13,2025-10-16
l13,p7,u2,2025-10-21
l14,p10,u11,2025-10-23
l15,p4,u15,2025-09-28
```

`friendships.csv` **(relacionamento simétrico; criaremos** `[:FRIENDS]` **bidirecional no cypher)**

```
user_a,user_b,since,strength
u1,u2,2019-05-10,0.8
u1,u3,2020-11-02,0.4
u2,u7,2018-07-14,0.6
u2,u12,2021-02-20,0.7
u3,u11,2023-06-01,0.5
u4,u16,2017-09-15,0.9
u5,u13,2016-10-20,0.95
u6,u9,2022-01-12,0.3
u7,u14,2020-12-30,0.6
u8,u12,2024-03-05,0.4
u9,u15,2015-04-18,0.7
u10,u2,2019-01-01,0.5
u11,u17,2023-09-10,0.6
u12,u14,2022-11-11,0.65
u13,u5,2018-02-02,0.85
u14,u1,2021-08-08,0.55
u15,u5,2014-06-06,0.9
u16,u4,2017-09-15,0.9
u17,u3,2024-01-20,0.35
u18,u2,2020-04-04,0.45
u19,u5,2022-07-01,0.5
u20,u10,2010-10-10,0.6
```

`memberships.csv` **(usuários em grupos)**

```
user_id,group_id,role,joined_at
u2,g1,moderator,2024-01-15
u14,g1,member,2024-02-01
```

```
u7,g1,member,2024-03-10
u12,g2,member,2023-11-06
u6,g2,member,2023-11-06
u1,g2,member,2024-05-01
u5,g3,admin,2022-09-01
u13,g3,member,2022-09-05
u16,g4,moderator,2023-04-01
u4,g4,member,2023-04-02
u11,g5,admin,2024-06-01
u17,g5,member,2024-06-15
u19,g3,member,2022-10-01
```

Script Cypher para criar constraints (execute antes de carregar)

```
CREATE CONSTRAINT user_id_unique IF NOT EXISTS
FOR (u:User) REQUIRE u.user_id IS UNIQUE;
CREATE CONSTRAINT post_id_unique IF NOT EXISTS
FOR (p:Post) REQUIRE p.post_id IS UNIQUE;
CREATE CONSTRAINT group_id_unique IF NOT EXISTS
FOR (g:Group) REQUIRE g.group_id IS UNIQUE;
CREATE CONSTRAINT comment_id_unique IF NOT EXISTS
FOR (c:Comment) REQUIRE c.comment_id IS UNIQUE;
```

Script Cypher para carregar CSVs (LOAD CSV)

> Coloque os arquivos no diretório `import/` do Neo4j e execute esses comandos no Browser ou via cypher-shell.

```
// USERS
LOAD CSV WITH HEADERS FROM 'file:///users.csv' AS row
MERGE (u:User {user_id: row.user_id})
SET u.name = row.name, u.age = toInteger(row.age), u.city = row.city,
    u.occupation = row.occupation, u.interests = row.interests;

// GROUPS
LOAD CSV WITH HEADERS FROM 'file:///groups.csv' AS row
MERGE (g:Group {group_id: row.group_id})
SET g.name = row.name, g.topic = row.topic, g.created_at = row.created_at;

// POSTS
LOAD CSV WITH HEADERS FROM 'file:///posts.csv' AS row
MERGE (p:Post {post_id: row.post_id})
SET p.content = row.content, p.created_at = row.created_at;
// relaciona autor -> post
MATCH (u:User {user_id: row.author_id}), (p:Post {post_id: row.post_id})
```

```
MERGE (u)-[:POSTED]->(p);

// COMMENTS
LOAD CSV WITH HEADERS FROM 'file:///comments.csv' AS row
MERGE (c:Comment {comment_id: row.comment_id})
SET c.content = row.content, c.created_at = row.created_at;
MATCH (u:User {user_id: row.author_id}), (p:Post {post_id: row.post_id}),
(c:Comment {comment_id: row.comment_id})
MERGE (u)-[:COMMENTED]->(c)
MERGE (c)-[:ON_POST]->(p);

// LIKES (posts)
LOAD CSV WITH HEADERS FROM 'file:///likes.csv' AS row
MATCH (u:User {user_id: row.user_id}), (p:Post {post_id: row.post_id})
MERGE (u)-[l:LIKED {created_at: row.created_at}]->(p);

// FRIENDSHIPS -> cria relacionamento bidirecional
LOAD CSV WITH HEADERS FROM 'file:///friendships.csv' AS row
MATCH (a:User {user_id: row.user_a}), (b:User {user_id: row.user_b})
MERGE (a)-[r:FRIENDS]->(b)
SET r.since = row.since, r.strength = toFloat(row.strength)
MERGE (b)-[r2:FRIENDS]->(a)
SET r2.since = row.since, r2.strength = toFloat(row.strength);

// MEMBERSHIPS
LOAD CSV WITH HEADERS FROM 'file:///memberships.csv' AS row
MATCH (u:User {user_id: row.user_id}), (g:Group {group_id: row.group_id})
MERGE (u)-[m:MEMBER_OF]->(g)
SET m.role = row.role, m.joined_at = row.joined_at;
```

Consultas de exemplo (insights)

1) Usuários recomendados (friends-of-friends) — sugestões de 1º grau indireto ordenadas por número de amigos em comum

```
MATCH (me:User {user_id: 'u1'})-[:FRIENDS]->(f)-[:FRIENDS]->(fof)
WHERE NOT (me)-[:FRIENDS]->(fof) AND me <> fof
RETURN fof.user_id AS recommended, fof.name AS name, count(DISTINCT f) AS
mutual_friends
ORDER BY mutual_friends DESC
LIMIT 5;
```

2) Post mais curtido no último mês (considerando data no formato YYYY-MM-DD)

```
WITH date() AS today
WITH date() - duration({months:1}) AS one_month_ago
MATCH (p:Post)<-[:LIKED]-(u:User)
```

```
WHERE date(p.created_at) >= one_month_ago
RETURN p.post_id, p.content, count(*) AS likes
ORDER BY likes DESC
LIMIT 5;
```

3) Quem curtiu o post mais curtido (e se são amigos entre si)

```
MATCH (p:Post)<-[:LIKED]-(u:User)
WITH p, count(u) AS likes
ORDER BY likes DESC
LIMIT 1
MATCH (p)<-[:LIKED]-(u:User)
OPTIONAL MATCH (u)-[:FRIENDS]-(other:User)
WHERE other IN collect(u)
RETURN p.post_id, u.user_id, u.name, size((u)-[:FRIENDS]-()) AS friends_count
LIMIT 50;
```

4) Menor caminho entre dois usuários (ex.: u1 -> u19)

```
MATCH (a:User {user_id:'u1'}), (b:User {user_id:'u19'})
CALL apoc.algo.dijkstra(a, b, 'FRIENDS', 'strength') YIELD path, weight
RETURN path, weight
LIMIT 1;
```

Se você não tiver APOC/GDS instalado, use `shortestPath()` simples:

```
MATCH (a:User {user_id:'u1'}), (b:User {user_id:'u19'})
MATCH p = shortestPath((a)-[:FRIENDS*1..5]-(b))
RETURN p
LIMIT 1;
```

5) Top autores por curtidas recebidas

```
MATCH (author:User)-[:POSTED]->(p:Post)<-[:LIKED]-(u:User)
RETURN author.user_id, author.name, count(*) AS total_likes
ORDER BY total_likes DESC
LIMIT 10;
```

6) Comunidades (usando o algoritmo Louvain do Graph Data Science)

```
// exemplo com GDS
CALL gds.graph.create('socialGraph', ['User','Post','Group','Comment'],
{FRIENDS:{}, POSTED:{}, MEMBER_OF:{}, LIKED:{}})
CALL gds.louvain.stream('socialGraph')
YIELD nodeId, communityId
```

```
RETURN gds.util.asNode(nodeId).user_id AS user_id, communityId
ORDER BY communityId, user_id
LIMIT 100;
```

Dicas de uso

- Se usar `LOAD CSV` no Browser, coloque os CSVs em `neo4j/import/` (ou configure `dbms.directories.import` no `neo4j.conf`).
- Para conjuntos maiores, prefira `neo4j-admin import` (mais rápido) ou procedimentos em lote (APOC) para evitar transações muito grandes.
- Ajuste tipos (por exemplo, converter `created_at` para `date(p.created_at)` quando necessário) e índices para performance.

Próximos passos (sugestões para o desafio)

- Expandir o dataset gerando mais usuários e interações (usar scripts Python para criar centenas de milhares de linhas se quiser simular carga real).
- Rodar algoritmos GDS: PageRank, Louvain, Node2Vec para recomendações baseadas em embeddings.
- Construir queries que respondam perguntas específicas do enunciado: alcance (reach), centralidade (degree, betweenness), influência (PageRank), e caminhos mínimos.