



# UNINTER

## ATIVIDADE PRÁTICA

### NoSQL

# SUMÁRIO

<b>SUMÁRIO .....</b>	<b>1</b>
<b>INTRODUÇÃO .....</b>	<b>2</b>
<b>MOTIVAÇÃO DO TRABALHO.....</b>	<b>2</b>
<b>RESUMO DO QUE FAZER .....</b>	<b>3</b>
<b>ORIENTAÇÕES GERAIS.....</b>	<b>6</b>
<b>FORMATO DE ENTREGA .....</b>	<b>6</b>
<b>CRITÉRIOS DE AVALIAÇÃO.....</b>	<b>7</b>
FORMATO DA APRESENTAÇÃO.....	8
IDENTIFICAÇÃO PESSOAL.....	9
CÓDIGO/QUERY .....	10
IMAGENS.....	11
RESPOSTA.....	12
<b>EXEMPLO DE APRESENTAÇÃO DE QUESTÃO .....</b>	<b>13</b>
<b>PRÁTICAS.....</b>	<b>14</b>
DESCRÍÇÃO DO CONJUNTO DE DADOS E PROJETO .....	14
DETALHAMENTO DOS PASSOS A SEREM REALIZADOS NO TRABALHO .....	18
CRIAÇÃO DE UM NOVO DBMS LOCAL EM VERSÃO 4.* .....	18
TESTE DO DBMS CRIADO E FUNCIONANDO COM A BIBLIOTECA APOC.....	41
EXEMPLO DE COMANDO DE CRIAÇÃO DE NÓS E RELACIONAMENTOS COM BASE NOS ARQUIVOS JSON DA PASTA	
IMPORT .....	43
QUESTÃO 01: IMPORTAÇÃO DOS ARQUIVOS JSON E CRIAÇÃO DE NÓS E ARESTAS COM BASE NOS DADOS DOS	
ARQUIVOS .....	48
QUESTÃO 02: DESCOBERTA DA HASHTAG PRINCIPAL .....	55
QUESTÃO 03: ANÁLISE DOS DADOS SEGUNDO VIÉS A SUA ESCOLHA.....	58
<b>RESPOSTAS AS DÚVIDAS MAIS FREQUÊNTES .....</b>	<b>60</b>

# INTRODUÇÃO

Olá a todos.

Sejam todos muito bem-vindos!

Esta avaliação foi planejada e preparada para a disciplina de Banco de Dados NoSQL dos Cursos de Tecnologia do Centro Universitário Internacional Uninter.

O objetivo desta atividade é fazer com que você, aluno, desenvolva os conhecimentos teóricos aprendidos na rota de aprendizagem, de maneira prática e aplicável no mercado de trabalho. Para tanto, será necessário o uso das ferramentas apresentadas nas aulas práticas, em especial na aula prática de Neo4j.

Ao longo desse roteiro serão passadas as orientações gerais para realização da avaliação bem como os seus critérios de correção. Na sequência, apresenta-se um exemplo comentado de como se deve ser entregue uma questão. Seguindo o roteiro estarão as práticas a serem realizadas, cada uma delas possui uma explicação de como deve ser feita, como será cobrada e algumas dicas. Por fim, apresento uma seção com as respostas das dúvidas mais frequentes realizadas por vocês. Bons estudos!

*No mais, desejo-lhe boa atividade prática em nome dos professores  
da disciplina de Banco de Dados NoSQL.*

As práticas desse roteiro utilizam 547 arquivos JSON (sugerimos usar de 3 a 10 arquivos em sequência apenas) com dados de Tweets coletados entre 31/12/2019 e 01/04/2021 com apenas uma hashtag como filtro para as mensagens. Sua tarefa é descobrir qual é esta hashtag (apenas ela aparece em todas os tweets originais coletados, excluindo-se os retweets, mensagens de citação e mensagens de resposta) e escolher algum viés de análise de relacionamento entre os dados coletados para apresentação em um grafo com os nós como os usuários ou os tweets e os relacionamentos como retweets ou citações ou outra forma de relacionamento que você perceba como relevante.

O desafio principal é a aplicação de todos os seus conhecimentos adquiridos até o momento para a realização de uma análise de dados com aplicação em banco orientado a grafos, bem como uso de seu senso crítico na escolha do que analisar e de qual janela de tempo analisar o dado em questão.

## MOTIVAÇÃO DO TRABALHO

Análise de dados de redes sociais é o grande consumidor de técnicas e aplicações de bancos de dados NoSQL. Sendo assim, treinar com uma aplicação em situação real de mercado é de suma importância.

Saber separar suas opiniões e convicções de sua análise técnica e imparcial de um dado é o que o tornará valorizado como excelente profissional.

## RESUMO DO QUE FAZER

1. **Instalar** Neo4j localmente ou em nuvem
2. **Criar** banco de dados vazio (**DBMS Local** em versão 4.\* sugerida)
3. Instalar biblioteca APOC neste DBMS criado
4. Abra a pasta import do DBMS criado e coloque nela de 3 a 10 arquivos JSON, em sequência numérica, dos 547 arquivos fornecidos, que contém os tweets e metadados coletados a serem analisados. (sugestão para não causar demora na importação, mas você pode usar todos os arquivos JSON se desejar)
5. Crie ou copie dos arquivos do trabalho em sua rota o arquivo apoc.conf na pasta conf do seu DBMS novo, para permitir acesso aos arquivos da pasta *import*.
6. Inicie seu DBMS e abra o Browser dele para realizar queries Cypher nele.
7. **Carregar dados** dos arquivos JSON da pasta import usando uma query Cypher para criação dos **nós** e **relacionamentos** no seu DBMS vazio (nós e arestas com importação de dados dos arquivos JSON da pasta import - **incluir o dado de referenced\_tweets.type nos nós de Tweet**). Observações: Não esqueça de explicar o funcionamento destes comandos na sua legenda. Dicas e comandos sugeridos (necessário adaptar e complementar de acordo com sua necessidade):
  - a. Colocar o dado de **referenced\_tweets.type**
  - b. Colocar demais dados que serão necessários para a última questão (ver item 7)
  - c. `MERGE (identificador:RU {ru: "RU-12345678"})`;
  - d. 

```
CALL apoc.load.directory("*.json") YIELD value
WITH value as arquivo
CALL apoc.load.json(arquivo) YIELD value
UNWIND value.data as tweet
MERGE (u:User {user_id: tweet.author_id})
MERGE (t:Tweet {id_tuite: tweet.id})
ON CREATE SET t += {
  //... (Coloque aqui os dados que você deseja)
}
//... (Coloque aqui os demais nós e relacionamentos que você deseja criar)
FOREACH (hash IN tweet.entities.hashtags |
  MERGE (h:Hashtag {tag:
    apoc.text.replace(apoc.text.clean(hash.tag), '[^a-zA-Z0-9]', '')})
  MERGE (h)-[:POSSUI]-(t)
  MERGE (u)-[:USOU]-(h)
)
FOREACH (ref_tweet IN tweet.referenced_tweets |
  SET t.tipo_ref = coalesce(t.tipo_ref, []) + [ref_tweet.type], t.id_ref
  = coalesce(t.id_ref, []) + [ref_tweet.id]
);
```
8. **Alterar os nós** de Tweet que contenham o dado referenced\_tweets na estrutura JSON para nós dos tipos dados em referenced\_tweets.type (estes nós não são Tweets e sim Retweets ou Quoted ou

Replied\_to e não podem influenciar na sua resposta da busca da hashtag principal). Comandos sugeridos:

- a. MATCH (t)
 

```

WHERE "retweeted" IN t.tipo_ref
REMOVE t:Tweet
SET t:Retweet;
```
- b. MATCH (t)
 

```

WHERE "replied_to" IN t.tipo_ref
REMOVE t:Tweet
SET t:Resposta;
```
- c. MATCH (t)
 

```

WHERE "quoted" IN t.tipo_ref
REMOVE t:Tweet
SET t:Citacao;
```

9. Fazer uma query de busca nos dados do seu banco para encontrar a hashtag principal, que aparece em todos os nós do tipo Tweet (não nos demais nós de mensagem não originais). Estilos de query sugeridos (necessário adaptar para seus dados):

- a. MATCH (n:NoImportante) - [r:RELACIONAMENTO\_QUALQUER] - (o:OutroNo)
 

```

WITH n, COUNT(r) AS contagem
ORDER BY contagem DESC
LIMIT 1
WITH n, contagem
MATCH (n)-[r:RELACIONAMENTO_QUALQUER]-(o:OutroNo), (ru:RU)
RETURN n, o, r, contagem, ru
LIMIT 15;
```
- b. MATCH (o:OutroNo)
 

```

WITH collect(o) AS listaDeMensagens
MATCH (n:NoImportante)
WHERE ALL(o IN listaDeMensagens
        WHERE NOT isEmpty([ (n)<- [r:RELACIONAMENTO_QUALQUER]-(o) | r ] ))
WITH n
MATCH (n)-[r:RELACIONAMENTO_QUALQUER]-(o:OutroNo), (ru:RU)
RETURN n, o, r, ru
LIMIT 15;
```

10. Gerar um grafo de visualização que contenha o nó principal como a hashtag encontrada e todos os demais tweets interligados nela (limitar visualização entre 10 e 20 nós contando com o central de hashtag. Os de mensagens e o seu nó de RU, se existir).

11. Fazer uma busca nos dados do seu banco em busca de alguma informação que você ache interessante a destacar. Sugestões de questões a serem respondidas (Algumas sugestões precisam ter a inclusão de dados nos atributos dos nós com base nos dados dos arquivos JSON e devem ser incluídos na criação do seu banco e não após):

- a. Qual o usuário que mais movimentou a rede de acordo com seus dados? (necessário criar nós de usuários e seus relacionamentos com os nós de mensagem)
  - b. Qual o equipamento mais usado para postar mensagens? (necessário criação de nós de equipamento e relacionamentos com as mensagens)
  - c. Quais as 3 hashtags menos usadas?
  - d. Qual o período (granularidade de hora) com maior movimentação da rede? (Necessário adicionar atributo com dado de created\_at)
12. Gerar um grafo de visualização que contenha o foco da sua informação buscada (limitar visualização entre 10 e 20. Não esquecer o seu nó de RU, se existir).
13. Abra o arquivo DOCX caderno de resposta e cole seus dados conforme solicitado.
14. Entregue o trabalho

# ORIENTAÇÕES GERAIS

## FORMATO DE ENTREGA

A entrega desta atividade prática deverá ser realizada pela área de “Trabalhos”, em formato PDF com o caderno de resolução da atividade prática.

Em seu caderno de resolução deverão constar as imagens referentes às duas questões, com o código/query usado para geração do resultado apresentado, o resultado da questão e mais o seu RU em algum lugar da imagem e do código (sua IP = Identificação Pessoal = seu RU).

O formato de entrega desejável dos prints das práticas desse roteiro, deve estar de acordo com o que é visto na seção “EXEMPLO DE APRESENTAÇÃO DE PRÁTICA”.

Recomenda-se que os trabalhos sejam enviados no formato .pdf. Uma vez que formatos .doc ou .docx podem apresentar falhas do tipo na codificação, carregamento ou apresentação de imagens. Sendo assim, fica **por conta e risco do estudante** se houver problemas com o documento enviados no formato doc ou docx ou outro formato editável.

## CRITÉRIOS DE AVALIAÇÃO

Os critérios de avaliação desse trabalho visam deixar a avaliação o mais justa e transparente possível. Nessa avaliação teremos um total de 100 pontos de trabalho, sendo 8 partes com valores iguais de 12,5 pontos cada divididas em 3 questões.

A primeira questão será composta por print do código e uma imagem com o resultado do código (resultado textual, sem geração de grafos ou tabelas). As segunda e terceira questões conterão 1 parte de código, 1 parte de imagem com um grafo e 1 parte de resposta. As questões serão avaliadas e corrigidas individualmente conforme a seguinte equação:

$$N = (FE) \frac{(IP).(\text{COD}_{criação} + \text{IMG}_{Q01PII}) + (IP).(\text{COD}_{análise1} + \text{IMG}_{Q02PII}) + \text{RESP}_{Q02PIII} + (IP).(\text{COD}_{análise2} + \text{IMG}_{Q03PII}) + \text{RESP}_{Q03PIII}}{8}$$

Em que:

*N (Nota da Questão): Nota total do trabalho, podendo variar de 0 até 100.*

*FE (Formato da Entrega): Nota do Formato de Entrega, podendo variar de 0 até 1.*

*IP (Identificação Pessoal): Nota Identificação Pessoal (seu RU), podendo variar de 0 até 1.*

*COD<sub>criação</sub> (Códigos de criação do banco): Nota do Código/Query usado para criar o banco de dados a serem colocados na Questão 01 Parte I, podendo variar de 0 até 100.*

*IMG<sub>Q01PII</sub> (Imagens da Questão 01 Parte II): Nota da(s) Imagem(s) com resultado esperado a serem colocadas na Questão 01 Parte II, podendo variar de 0 até 100.*

*COD<sub>análise1</sub> (Códigos de análise de dados do banco da Questão 02): Nota do Código/Query usado para criar o banco de dados a serem colocados na Questão 02 Parte I, podendo variar de 0 até 100.*

*IMG<sub>Q02PII</sub> (Imagens da Questão 02 Parte II): Nota da(s) Imagem(s) com resultado esperado a serem colocadas na Questão 02 Parte II, podendo variar de 0 até 100.*

*RESP<sub>Q02PIII</sub> (Resposta da Questão 02 Parte III): Nota da Resposta com resultado correto para a Questão 02 Parte III, podendo ser 0 ou 100.*

*COD<sub>análise2</sub> (Códigos de análise de dados do banco da Questão 03): Nota do Código/Query usado para criar o banco de dados a serem colocados na Questão 03 Parte I, podendo variar de 0 até 100.*

*IMG<sub>Q03PII</sub> (Imagens da Questão 03 Parte II): Nota da(s) Imagem(s) com resultado esperado a serem colocadas na Questão 03 Parte II, podendo variar de 0 até 100.*

*RESP<sub>Q03PIII</sub> (Resposta da Questão 03 Parte III): Nota da Resposta com resultado correto para a Questão 03 Parte III, podendo ser 0 ou 100.*

Cada um dos itens/critérios que compõe a equação acima será detalhado nas subseções a seguir.

**Se mesmo assim houver dúvidas, não hesite em perguntar. O desconhecimento dos critérios não será aceito como desculpa!**

## FORMATO DA APRESENTAÇÃO

O formato da apresentação é um dos critérios de avaliação, pois um profissional deve ser capaz de seguir normas no momento de elaboração de relatórios técnicos, manuais e outros documentos afins, bem como ser capaz de apresentar seus dados de forma limpa e comprehensível.

As possíveis notas desse critério são apresentadas na tabela a seguir:

Tabela 1: Possíveis notas no formato de apresentação

NOTA	DESCRIÇÃO NA DEVOLUTIVA	COMENTÁRIOS
1,00	Formato da apresentação está correto	Está de acordo com o exemplo (ver a seção “EXEMPLO DE APRESENTAÇÃO DE PRÁTICA” para maiores detalhes)
0,70	Formato da apresentação está parcialmente correto	Está muito próximo do exemplo, mas apresenta alguns erros
0,50	Formato da apresentação está incorreto	Não seguiu o exemplo.

## IDENTIFICAÇÃO PESSOAL

Todas as questões deverão apresentar um identificador pessoal nas seguintes partes:

- No código ou query deve haver ao menos uma variável cujo nome seja composto pelo seu RU (e.g. contadorxxxxxx – onde o “x” s deve ser substituído pelo seu RU), mesmo que esta variável não seja utilizada em nenhuma parte do código ou query.
- Nas imagens/resultados, onde deverá conter seu RU escrito em algum local.

As possíveis notas para esse critério são apresentadas na tabela a seguir:

Tabela 2: Possíveis notas critério de Identificação Pessoal

NOTA	DESCRÍÇÃO NA DEVOLUTIVA	COMENTÁRIOS
<b>1,00</b>	Apresentou o identificador pessoal no código/query e nas imagens/fotos.	Está de acordo com o exemplo (ver a seção “EXEMPLO DE APRESENTAÇÃO DE QUESTÃO” para maiores detalhes).
<b>0,80</b>	Apresentou identificador pessoal na imagem, mas não no código.	Não apresentou um identificador no código (e.g. o RU como parte do nome de uma variável)
<b>0,70</b>	Apresentou o identificador pessoal no código, mas não nas imagens/prints.	Não apresentou um identificador na imagem.
<b>0,50</b>	Não apresentou identificador pessoal no código/query e nem nas imagens/prints.	Questão sem nenhuma identificação de autoria.
<b>0,00</b>	Apresentou o identificador de outra pessoa nas prints e/ou no código/query.	A questão veio com identificador pessoal de outra pessoa.

## CÓDIGO/QUERY

A apresentação do código/query compõe um terço da nota total das questões. Este será avaliado conforme a tabela a seguir:

As possíveis notas para esse critério são apresentadas na tabela a seguir:

Tabela 3: Possíveis notas na apresentação do código

NOTA	DESCRIPÇÃO NA DEVOLUTIVA	COMENTÁRIOS
100	Código ou query <b>coerente com a resposta encontrada</b> e apresentado no formato <b>imagem</b> .	Está de acordo com o exemplo (ver a seção “EXEMPLO DE APRESENTAÇÃO DE QUESTÃO” para maiores detalhes)
70	Código ou query <b>coerente com a resposta encontrada</b> e apresentado no formato <b>texto</b> .	Acertou o código ou query, mas copiou o texto do código ao invés de tirar <i>print</i>
60	Código ou query <b>parcialmente correto</b> e apresentado no formato <b>imagem</b> .	Errou um pouco código ou query, mas colocou no trabalho no formato <b>imagem</b>
40	Código ou query <b>parcialmente correto</b> e apresentado no formato <b>texto</b> .	Errou um pouco código e copiou o texto do código ou query ao invés de tirar <i>print</i>
0	Sem código/query ou com código/query <b>incorrecto</b>	A questão não apresentou código/query ou o código/query estava errado.

**OBS. 1: NÃO ESQUECER DO IDENTIFICADOR PESSOAL (Ex.: COLOCAR SEU RU NO NOME DE UMA VARIÁVEL OU LOCAL DE SUA QUERY).**

## IMAGENS

As imagens compõem um terço da nota total de cada questão. Essas, normalmente, são prints dos grafos gerados pela sua busca na base de dados criada com os arquivos JSON fornecidos. Cada prática/questão dessa atividade prática virá com instruções de como devem ser essas imagens.

Entende-se que a **legenda faz parte de uma imagem**. Sendo assim, as **legendas serão avaliadas**.

As possíveis notas para esse critério são apresentadas na tabela a seguir:

Tabela 4: Possíveis notas na apresentação das imagens/fotos

NOTA	DESCRÍÇÃO NA DEVOLUTIVA	COMENTÁRIOS
100	Imagens <b>corretas</b> e com legenda <b>adequada</b> .	Está de acordo com o exemplo (ver a seção “EXEMPLO DE APRESENTAÇÃO DE QUESTÃO” para maiores detalhes)
90	Imagens <b>correta</b> , mas com legenda <b>superficial</b> .	Ex. de legenda superficial: “Figura 1: Nuvem de palavras”.
80	Imagens <b>corretas</b> , mas com legenda <b>precária</b> .	Ex. de legenda precária: “Figura 1: Imagem”
70	Imagens <b>correta</b> , mas <b>sem</b> legenda.	Apresentou imagens corretas, mas não colocou legenda.
60	Imagens <b>parcialmente</b> corretas, mas com legenda <b>adequada</b> .	Imagen que não consiga identificar o que esteja acontecendo ou a falta de uma das imagens se encaixam nesse grupo.
50	Imagens <b>parcialmente</b> correta, e com legenda <b>superficial</b> .	Similar ao segundo item de cima para baixo dessa tabela, mas com pelo menos uma das imagens com problemas.
40	Imagens <b>parcialmente</b> corretas, e com legenda <b>precária</b> .	Similar ao terceiro item de cima para baixo dessa tabela, mas com pelo menos uma das imagens com problemas.
30	Imagens <b>parcialmente</b> correta, e <b>sem</b> legenda.	Similar ao quarto item de cima para baixo dessa tabela, mas com pelo menos uma das imagens com problemas.
0	Sem imagens ou com imagens <b>incorrectas</b>	A questão veio sem imagens ou com imagens erradas

**OBS. 1: NÃO ESQUECER DO IDENTIFICADOR PESSOAL (Ex.: Colocar seu RU dentro da nuvem de palavras).**

## RESPOSTA

A apresentação da resposta correta será avaliada de forma booleana:

As possíveis notas para esse critério são 0 ou 100:

Tabela 5: Possíveis notas na apresentação das respostas

NOTA	DESCRIÇÃO NA DEVOLUTIVA	COMENTÁRIOS
100	Resposta <b>correta</b>	Está de acordo com o exemplo (ver a seção “EXEMPLO DE APRESENTAÇÃO DE QUESTÃO” para maiores detalhes)
0	Resposta <b>incorrecta</b>	A questão não apresentou a resposta correta para a pergunta.

# EXEMPLO DE APRESENTAÇÃO DE QUESTÃO

## Prática XX – Título da prática

### Questão XX – Enunciado curto da questão

**Enunciado:** Enunciado mais descriptivo sobre o que fazer na questão.

#### I. Apresentação do Código (não esquecer do identificador pessoal):

```

MATCH (n:NoImportante)-[r:RELACIONAMENTO_QUALQUER]-(o:OutroNo)
WITH n, COUNT(r) AS contagem
ORDER BY contagem DESC
LIMIT 1
WITH n, contagem
MATCH (n)-[r1:RELACIONAMENTO_QUALQUER]-(o:OutroNo)-[r2]-(p:NósExtra), (ru1234567:RU)
RETURN n, o, r1, p, r2, contagem, ru1234567
LIMIT 15;
    
```

Figura 1: Query de busca de dados no banco de dados já criado.

#### II. Apresentação das Imagens/Prints (não esquecer do identificador pessoal):

##### a. Grafo com o viés XYZ:

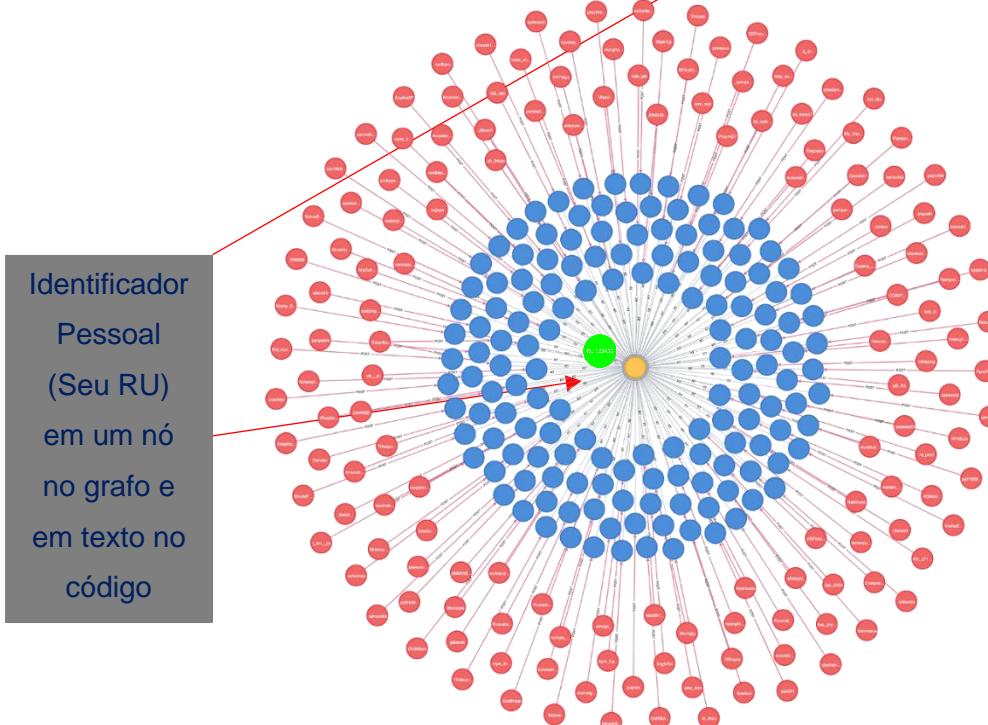


Figura 2: Resultado do grafo com os registros encontrados sobre o assunto XYZ. Ao centro vemos o tweet que gerou mais interação e em volta todos os tweets de interação como retweets, respostas, citações e outros.

OBS1: Nas suas imagens não precisa circular e apontar o identificador pessoal.

OBS2: Caso necessário, junte as imagens dos códigos, garantindo que não ocorra perda de clareza nem de organização.

OBS4: Este é apenas um exemplo e nem o código mostrado e nem o grafo são referentes à atividade prática que você deverá entregar, mas são muito próximos.

# PRÁTICAS

## DESCRIÇÃO DO CONJUNTO DE DADOS E PROJETO

Você deverá utilizar de 3 a 10 dos 547 arquivos disponibilizados e que são parte de um conjunto de mais de 1000 arquivos coletados entre 2006 e 2022 da plataforma X, antigo Twitter, usando como filtro apenas uma hashtag específica.

Cada arquivo contém aproximadamente 100 tweets que estão separados em 4 principais grandes contêineres de dados:

**data** – Principal fonte de dados para este projeto. Contém a mensagem e informações de contexto da mensagem, como nomes de usuários, hashtags usadas, se foi um retweet (dado em referenced\_tweets.type) ou não, se possui menção a outros usuários e muito mais.

**errors** – Contém detalhes sobre erros ocorridos durante a coleta dos dados, como usuários não encontrados ou deletados ou banidos ou suspensos. Estas podem não ser informações muito úteis neste trabalho. Nem todos os arquivos possuem estes dados.

**includes** – Contém dados sobre anexos, arquivos de mídia, links, informações extras sobre os tweets e informações extras sobre os usuários.

**meta** – Metadados sobre o arquivo JSON coletado. Aqui encontramos apenas 4 informações: id do tweet mais recente coletado, id do tweet mais antigo coletado, código token para coletar o próximo tweet da pesquisa e quantidade de mensagens contidas neste arquivo.

Para facilitar, segue estrutura padrão de um arquivo JSON deste trabalho:

### Estrutura geral:

```
1  {
2  >   "data": [ ...
7679 | ],
7680 >   "errors": [ ...
7782 | ],
7783 >   "includes": { ...
11488 | },
11489 >   "meta": { ...
11494 | }
11495 | }
```

[Estrutura básica de um tweet original \(veja que referenced\\_tweets não aparece em mensagens originais\):](#)

```
1  {
2      "data": [
3          {
4              "attachments": { ...
11         },
12         "author_id": "1233417213606055936",
13     },
14         "context_annotations": [ ...
73     ],
74         "conversation_id": "1241715364460998657",
75         "created_at": "2020-03-22T13:16:27.000Z",
76     },
77         "entities": { ...
145     },
146         "id": "1241715364460998657",
147         "lang": "pt",
148         "possibly_sensitive": false,
149     },
150         "public_metrics": { ...
154     },
155         "reply_settings": "everyone",
156         "source": "Twitter for iPhone",
157         "text": "A ... "
158     },
159 }
```

## Estrutura básica de um erro:

```
7455     "errors": [
7456     {
7457         "detail": "Could not find tweet",
7458         "parameter": "referenced_tweets.id",
7459         "resource_id": "1241693181344976",
7460         "resource_type": "tweet",
7461         "title": "Not Found Error",
7462         "type": "https://api.twitter.com",
7463         "value": "1241693181344976896"
7464     }.
```

**Estrutura básica de um include do tipo media (foto):**

```
7558     "includes": {  
7559         "media": [  
7560             {  
7561                 "height": 950,  
7562                 "media_key": "3_1241715458933510144",  
7563                 "type": "photo",  
7564                 "url": "https://pbs.twimg.com/media/E  
7565                     "width": 950  
7566             },
```

**Estrutura básica de um include do tipo place (lugar):**

```
7558     "includes": {  
7559         "places": [  
7560             {  
7561                 "country": "Brasil",  
7562                 "country_code": "BR",  
7563                 "full_name": "Bel\u00e3m, Brasil",  
7564                 "geo": {  
7565                     "bbox": [  
7566                         -48.623684,  
7567                         -1.526453,  
7568                         -48.295911,  
7569                         -1.019406  
7570                     ],  
7571                     "properties": {},  
7572                     "type": "Feature"  
7573                 },  
7574                 "id": "f3587bc643e7d7b8",  
7575                 "name": "Bel\u00e3m",  
7576                 "place_type": "city"  
7577             }  
7578         }  
7579     }  
7580 }
```

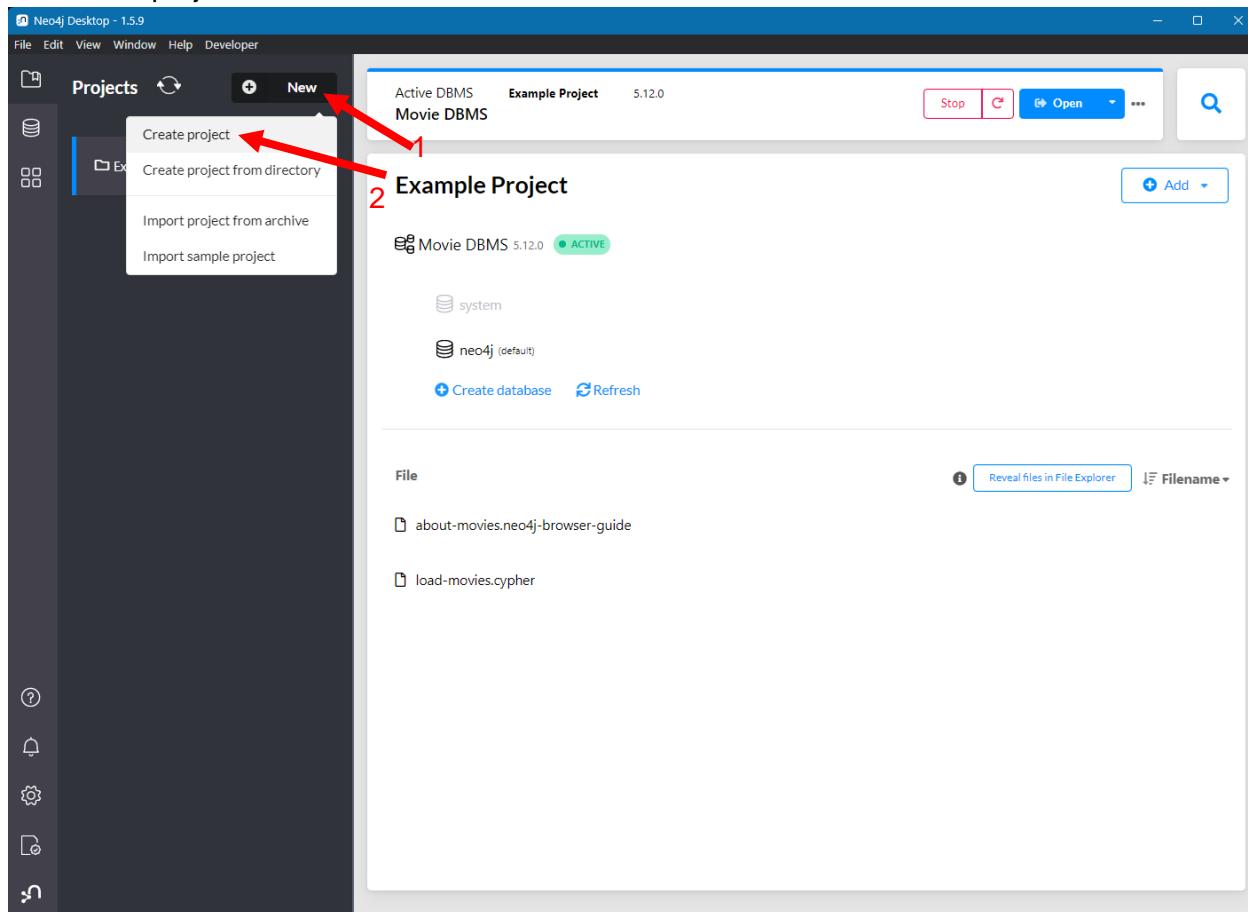
**Estrutura básica de um dado meta:**

```
11160     "meta": {  
11161         "newest_id": "1241715574029328384",  
11162         "next_token": "b26v89c19zqg8o3fo77ds",  
11163         "oldest_id": "1241479077158178817",  
11164         "result_count": 99  
11165     }  
11166 }
```

# DETALHAMENTO DOS PASSOS A SEREM REALIZADOS NO TRABALHO

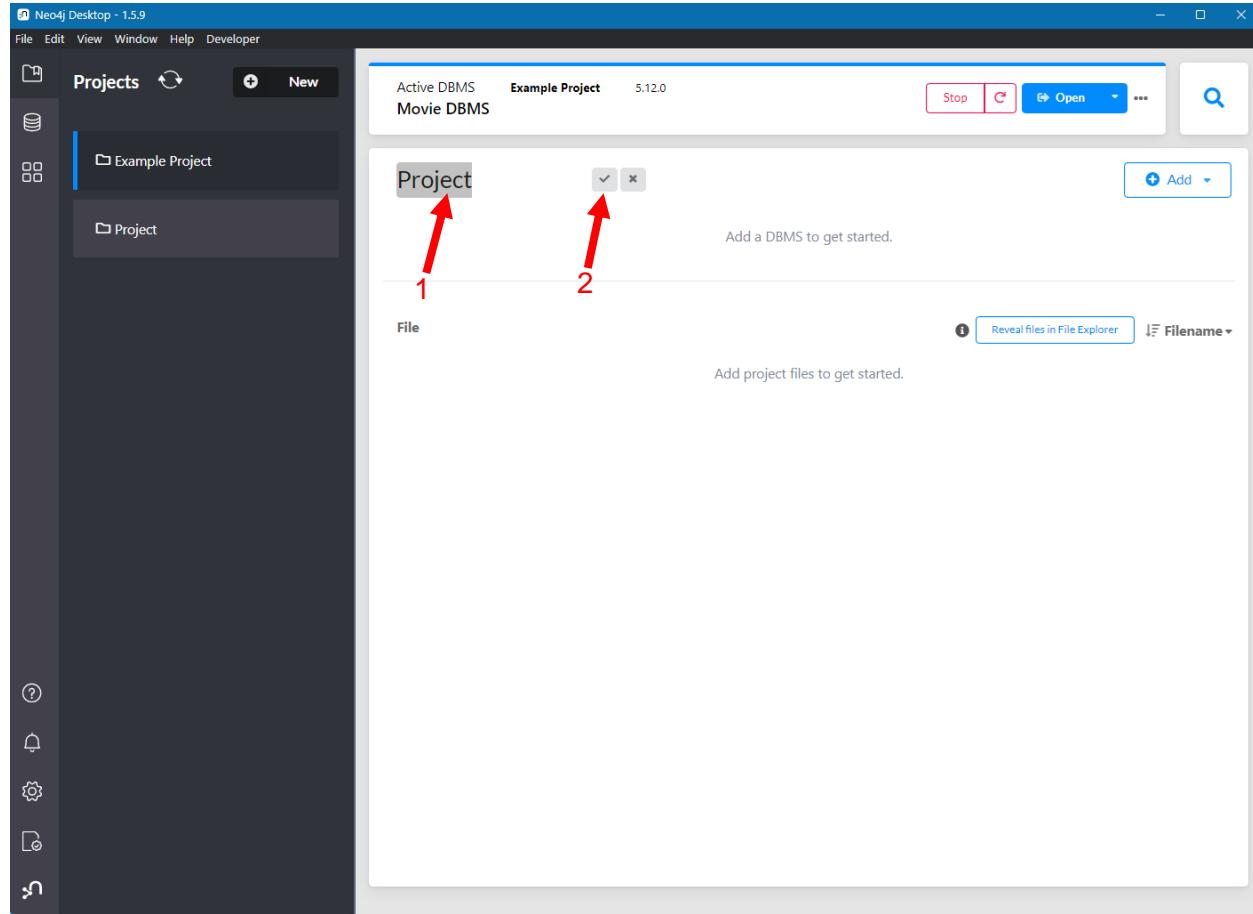
## CRIAÇÃO DE UM NOVO DBMS LOCAL EM VERSÃO 4.\*

### 1. Criar novo projeto:



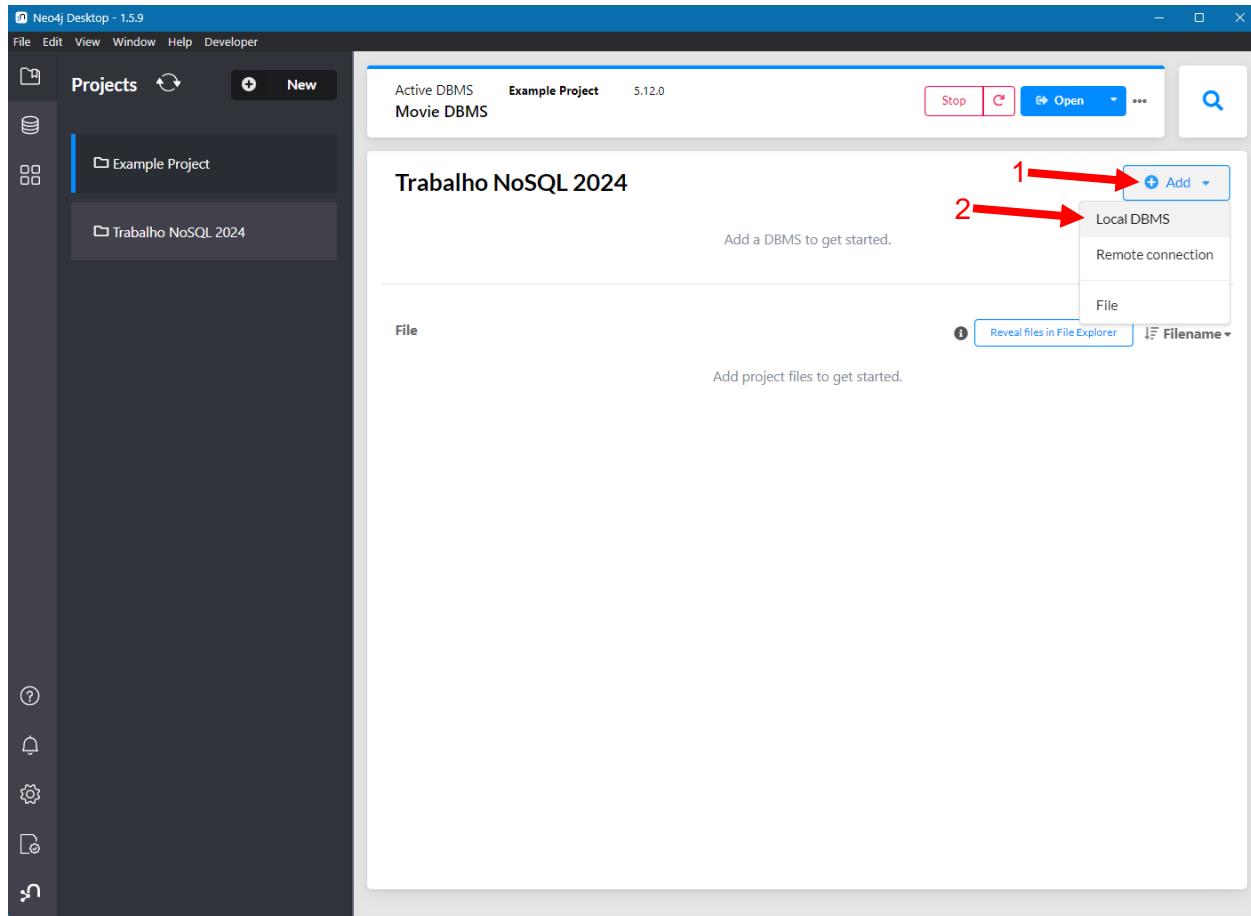


### 2. Renomear projeto:

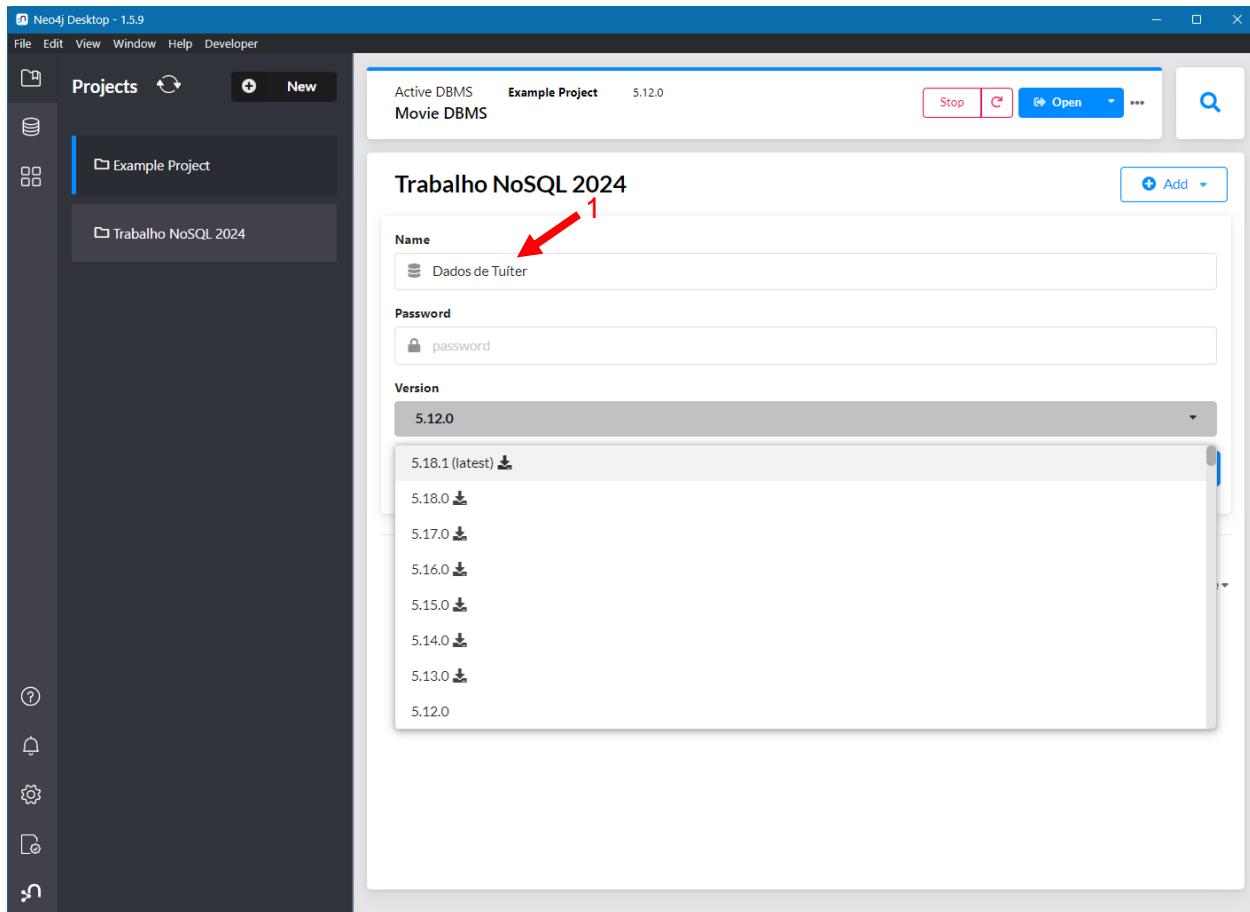




### 3. Adicionar novo DBMS local:



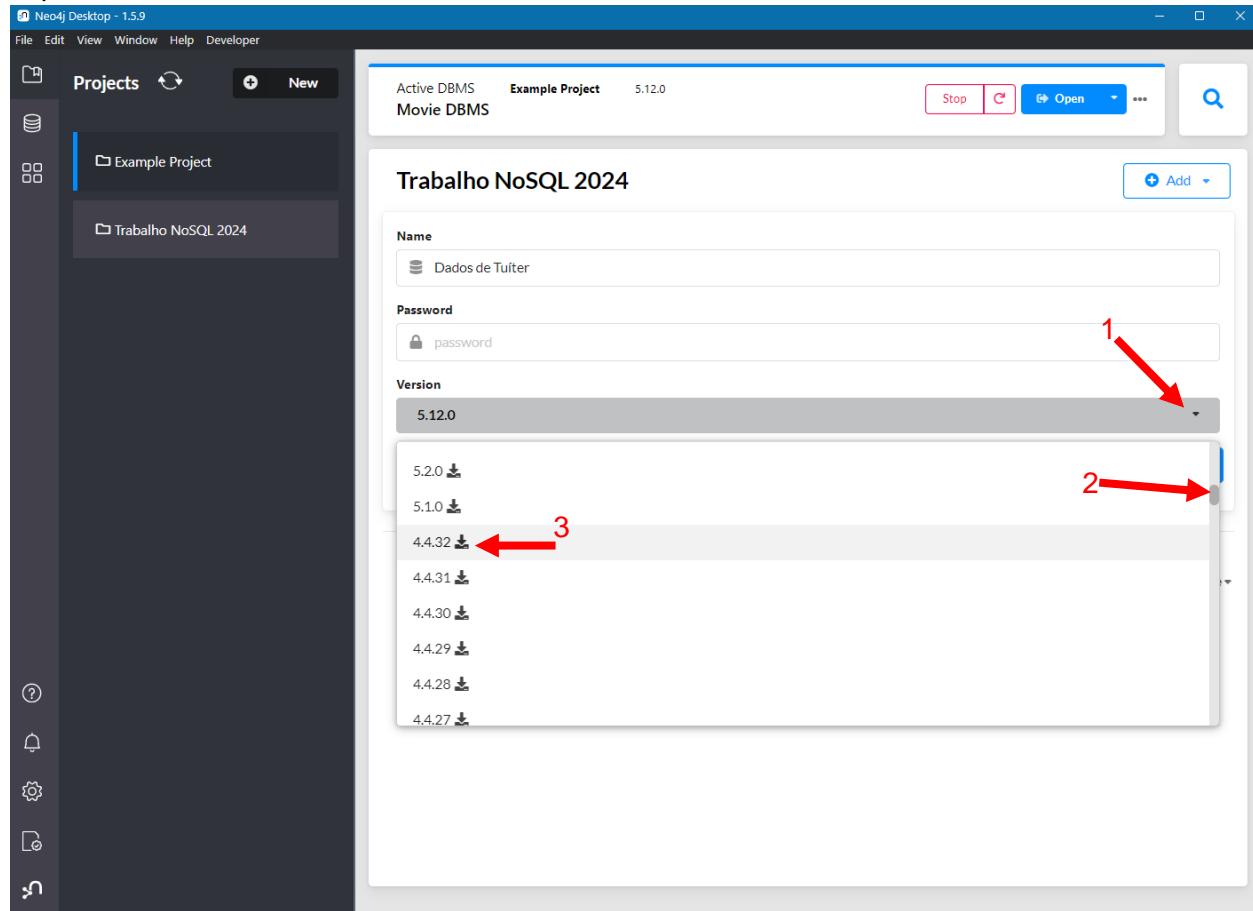
4. Renomear DBMS para algo que faça sentido para seu projeto e escolher a versão do DBMS a ser criado:



5. Sugerimos usar qualquer versão antes da 5, para ter acesso à biblioteca APOC por completo (após versão 5 a biblioteca APOC foi incorporada à base do CYPHER, porém alguns métodos não foram

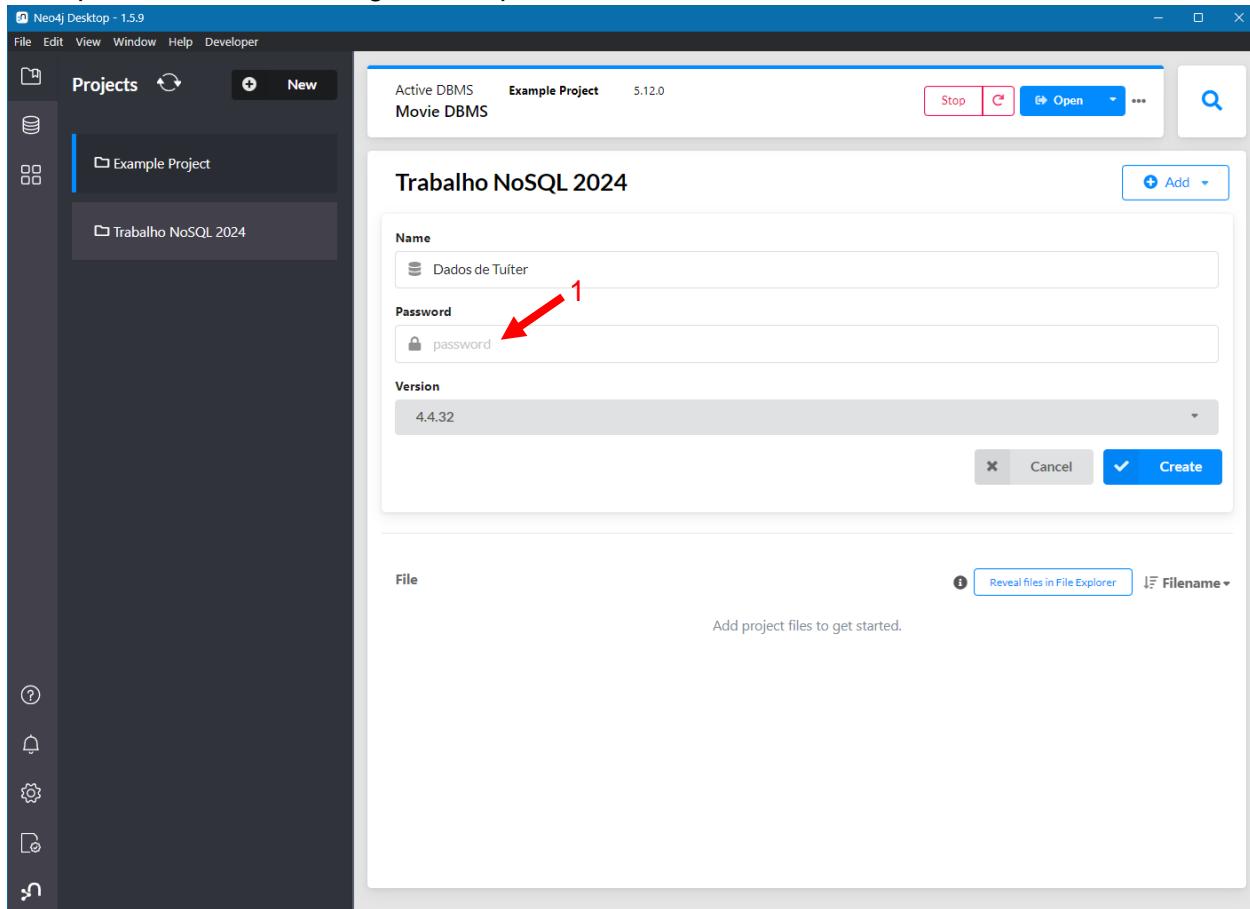


implementados ainda.



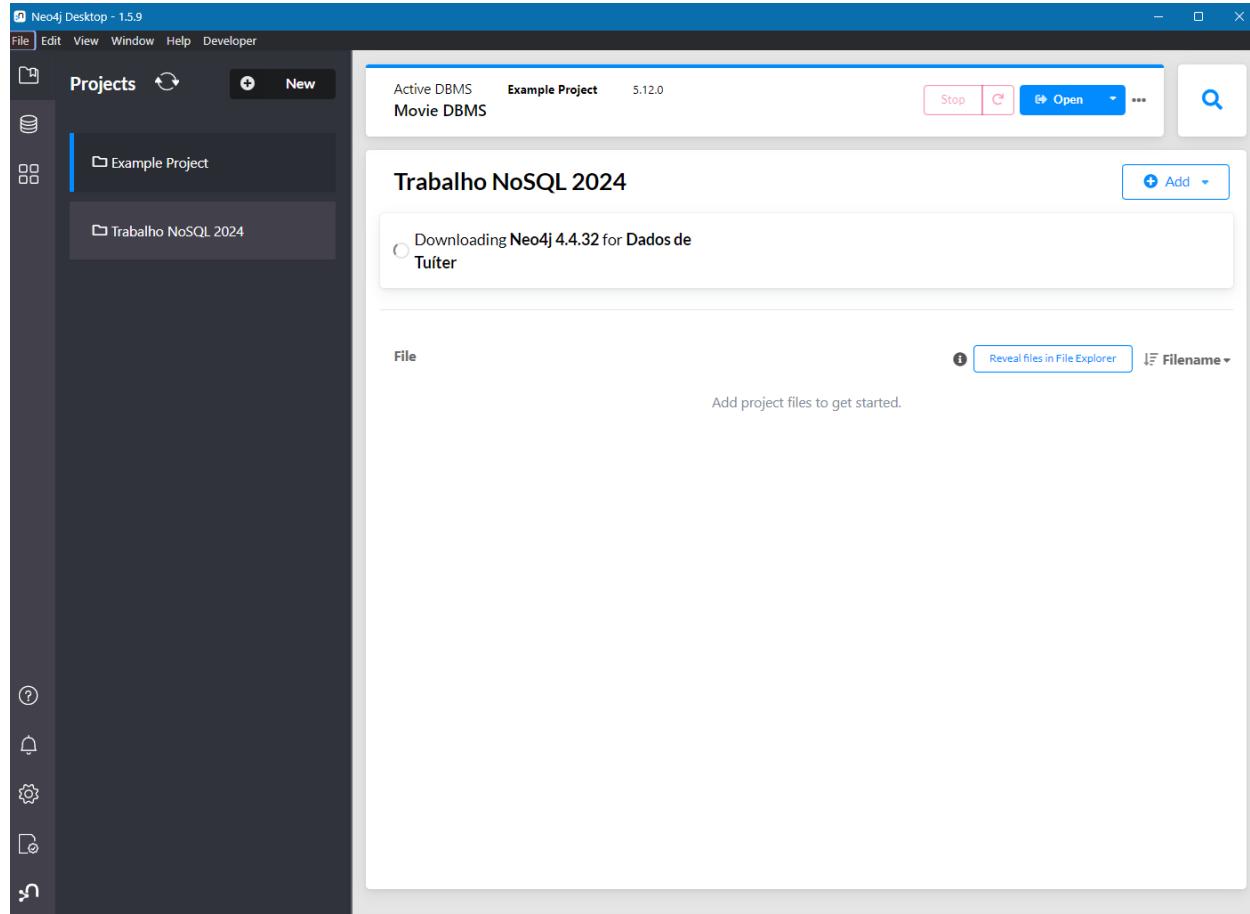


6. Coloque uma senha de 8 dígitos e clique em criar:

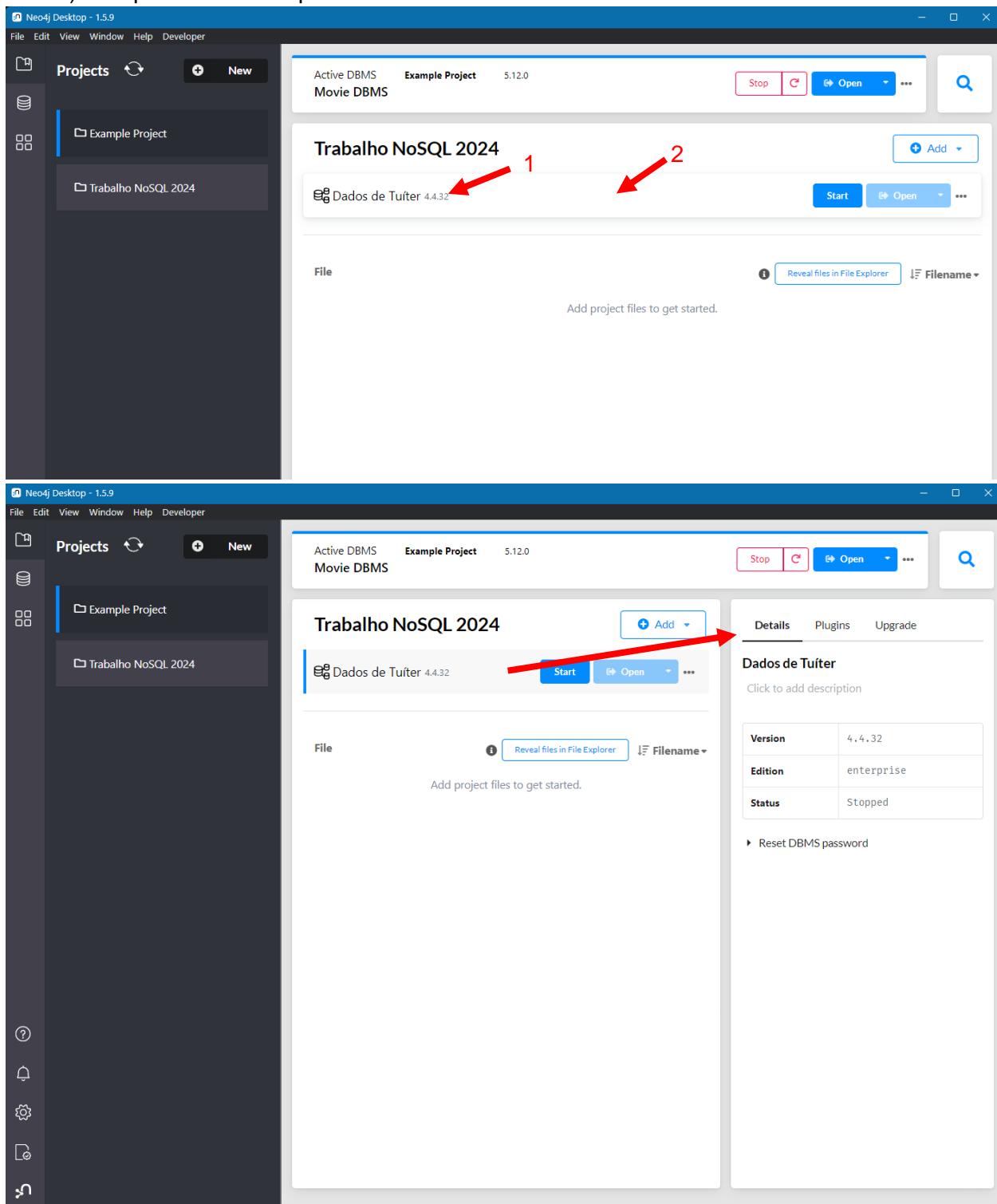




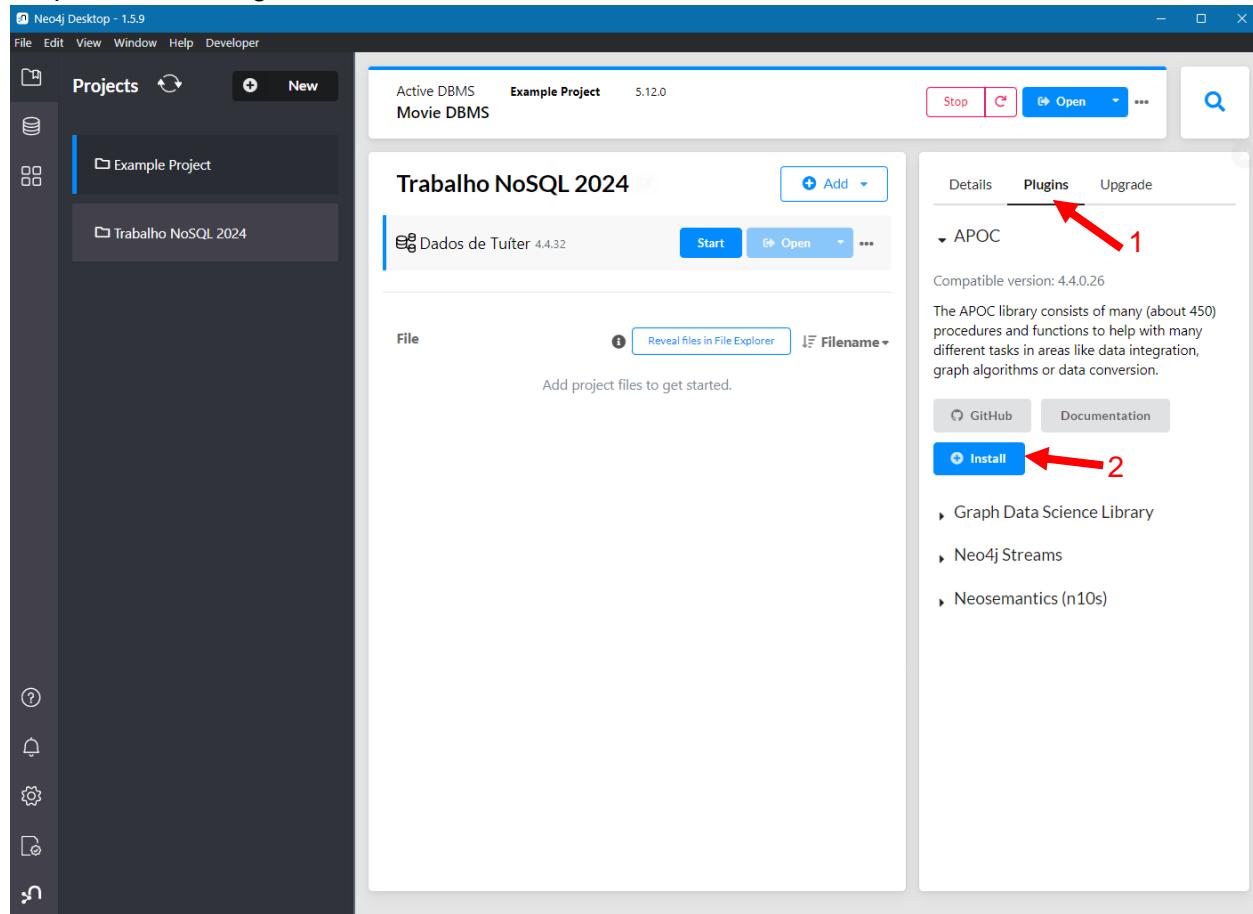
### 7. Aguarde o download e criação do banco:

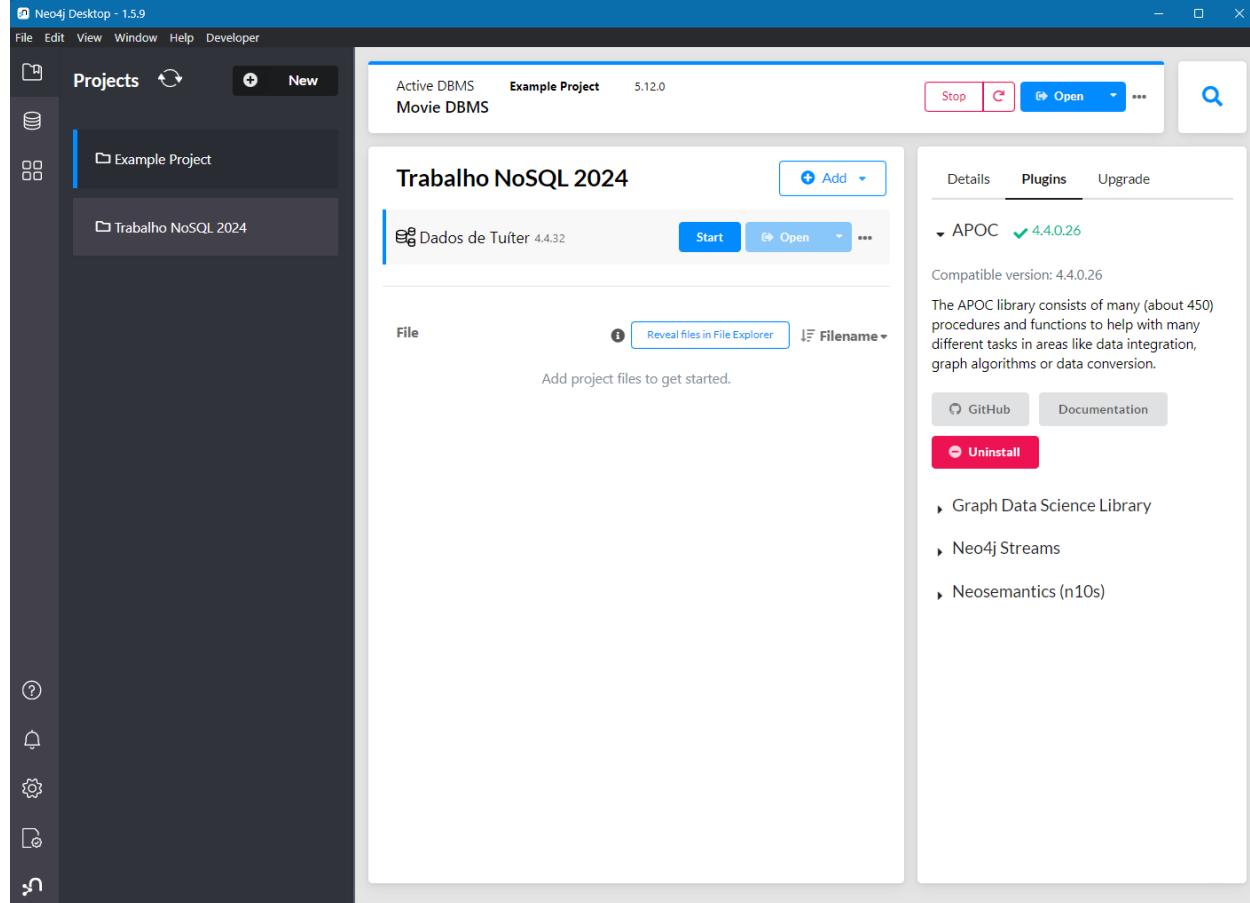
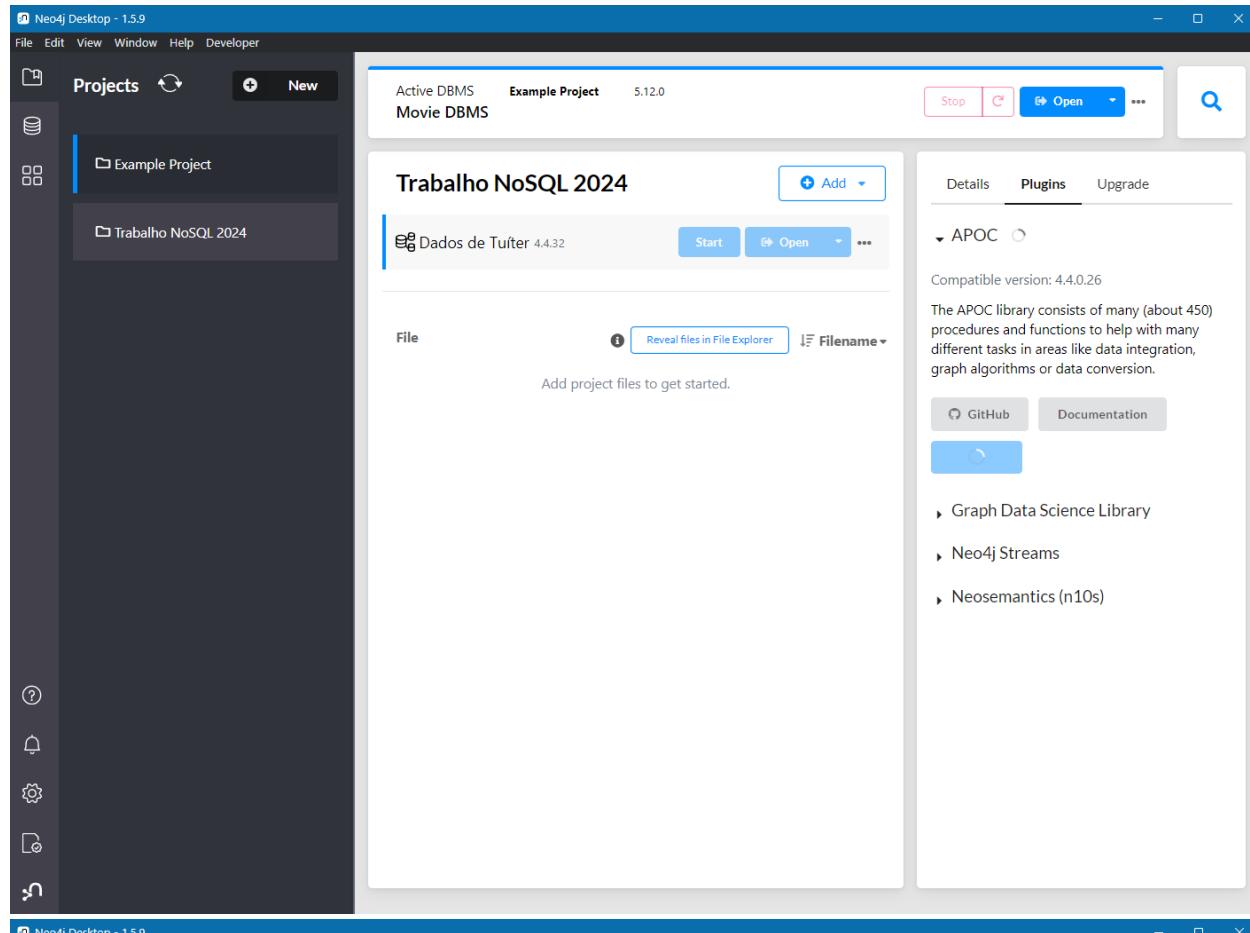


8. Com o banco criado, verifique se a versão está correta (número ao lado do nome do seu DBMS criado) e clique em cima da parte branca ao lado do número da versão:

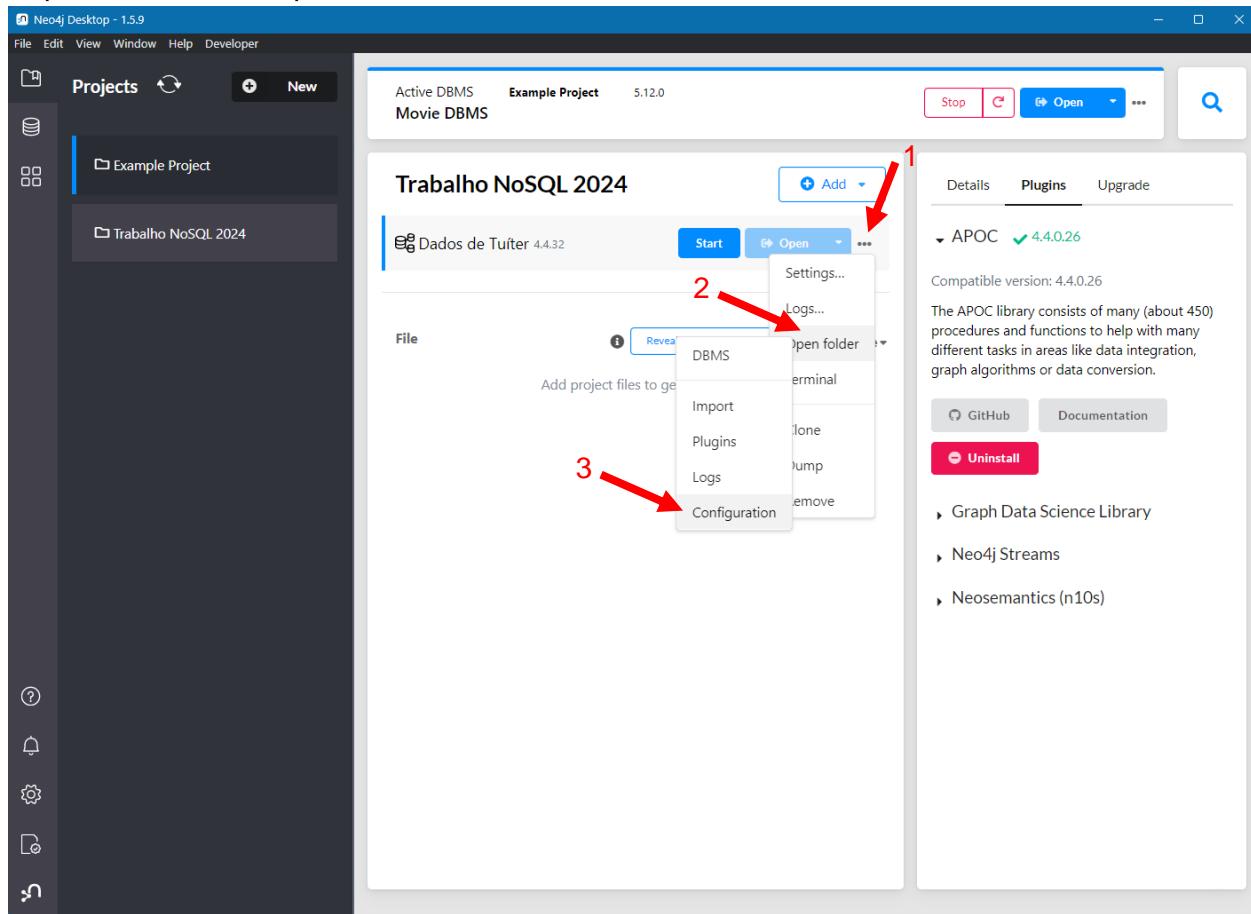


9. Clique na aba “Plugins” e instale a biblioteca APOC:

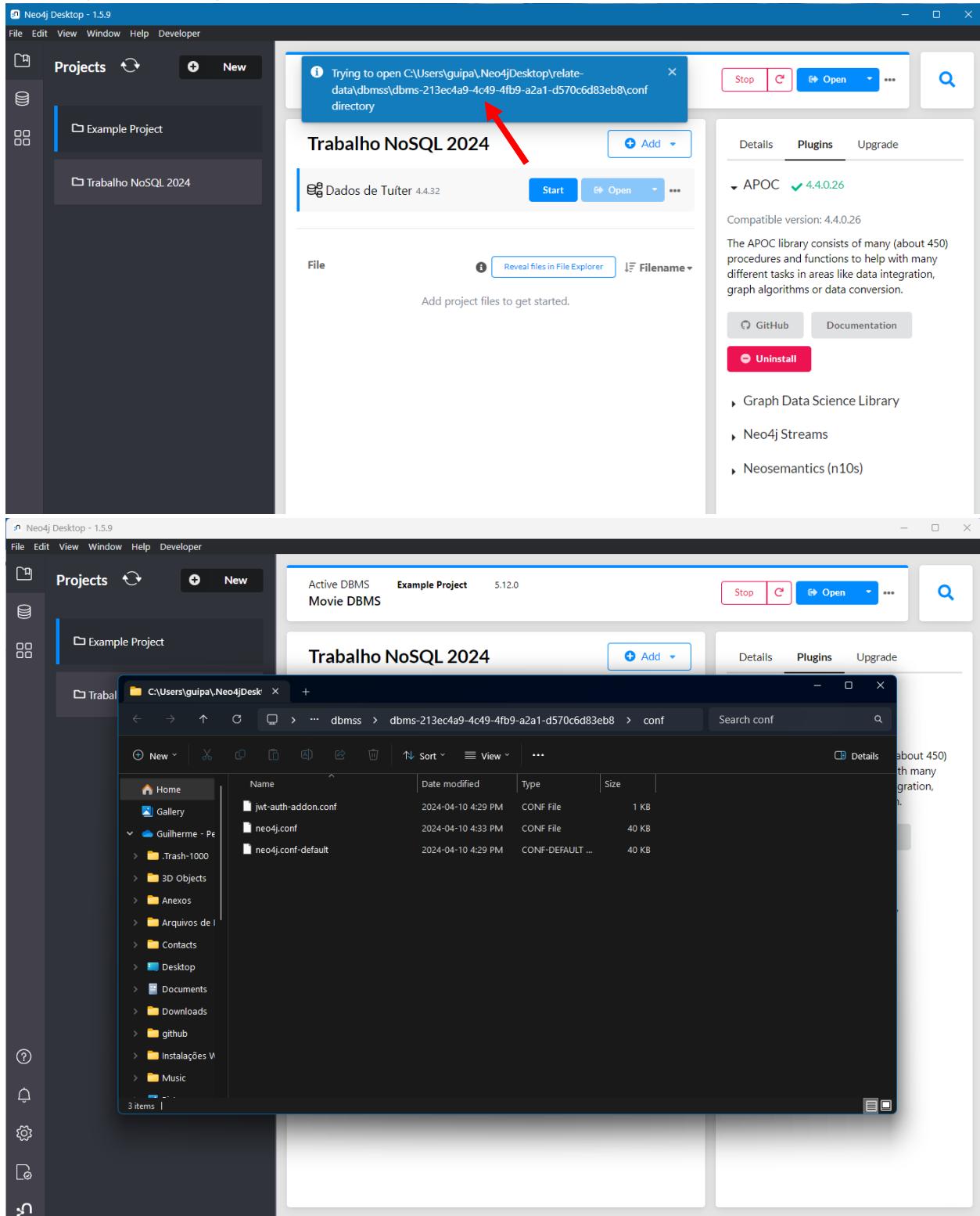




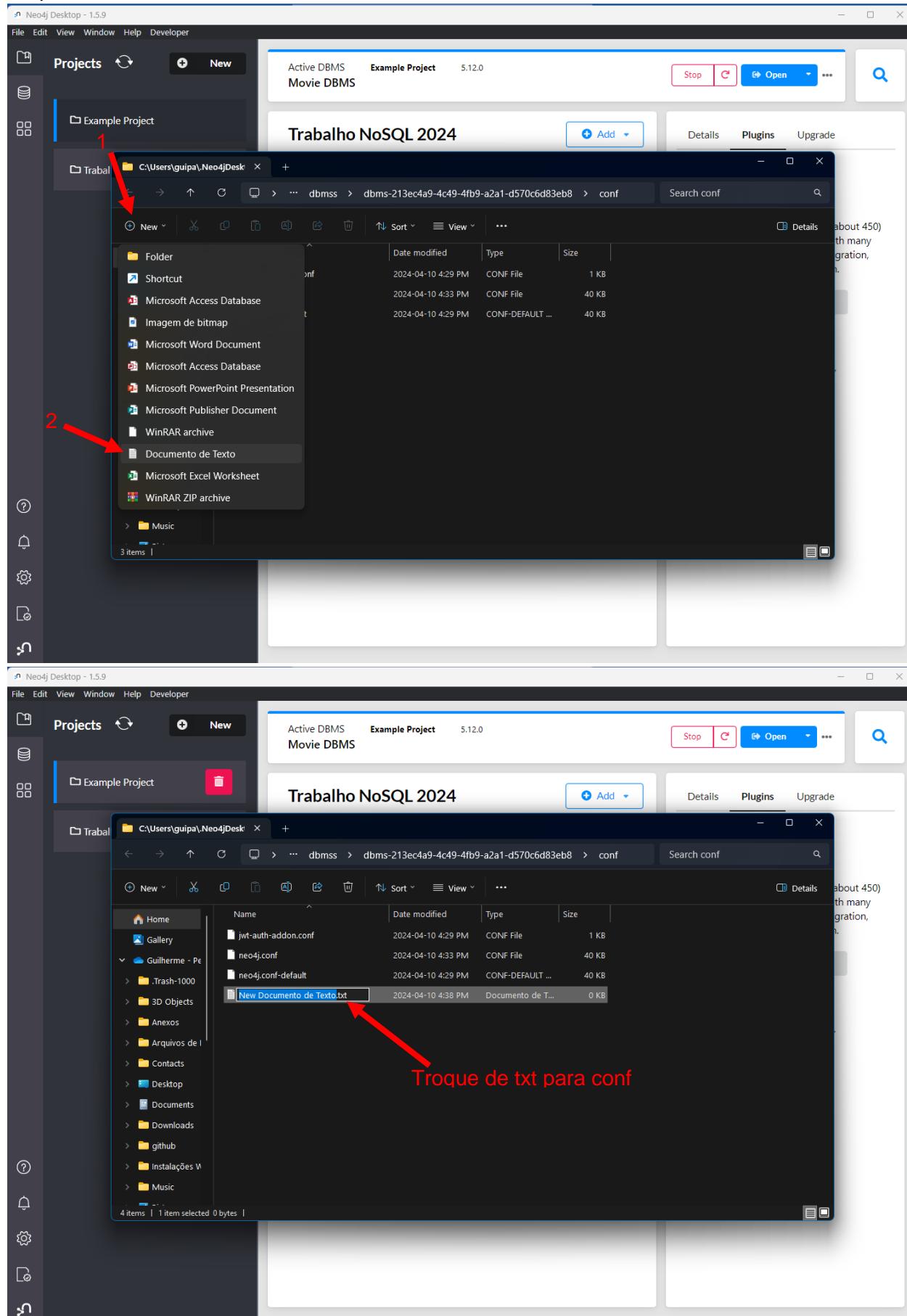
10. Clique nos três pontinhos ao lado do botão “Open” na barra com o nome do seu DBMS, clique em “Open folder” e clique em “Configuration” para abrir a pasta “Configuration” no seu gerenciador de arquivos do sistema operacional:



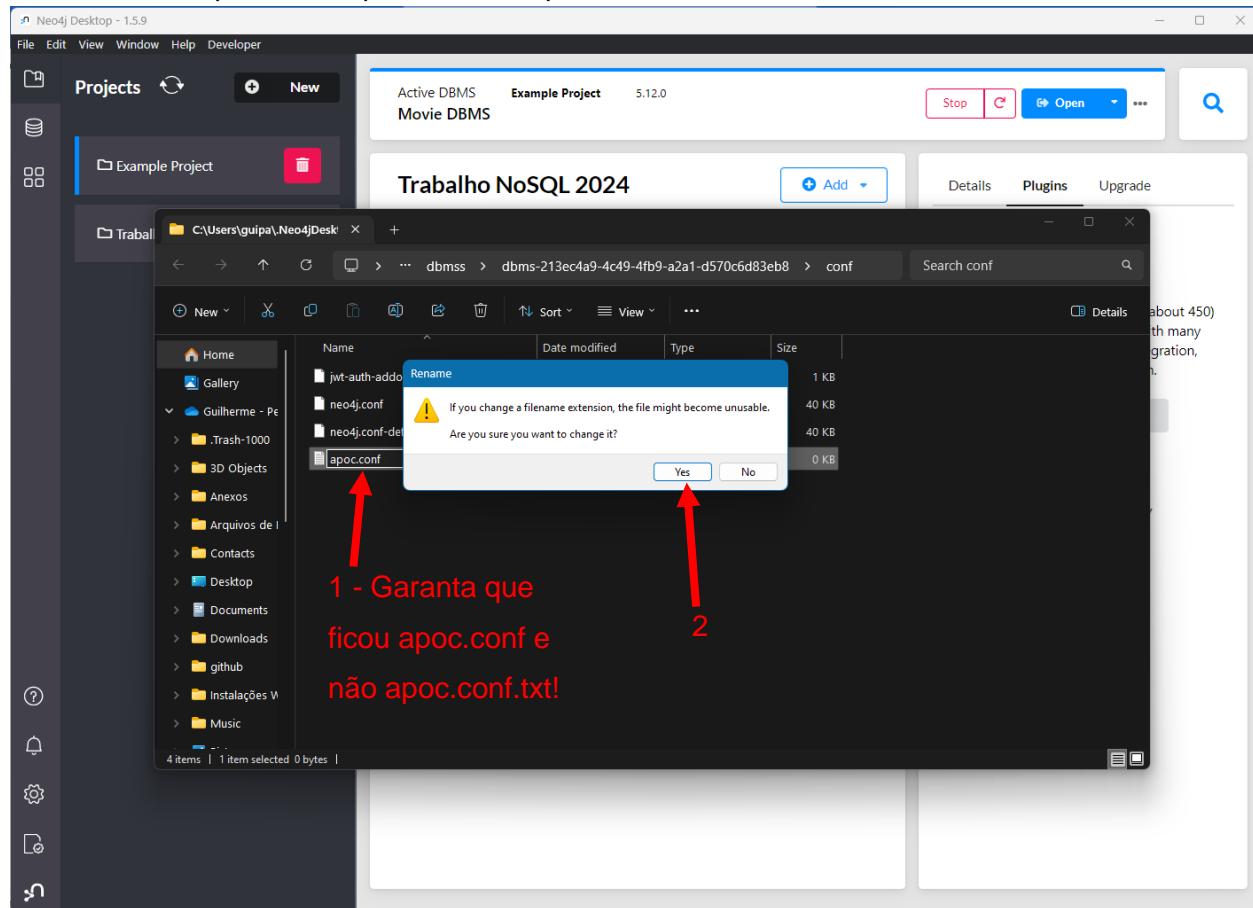
11. Uma mensagem em azul será mostrada, avisando que a pasta foi aberta. Use o ALT+TAB do seu teclado para ir até a pasta aberta:



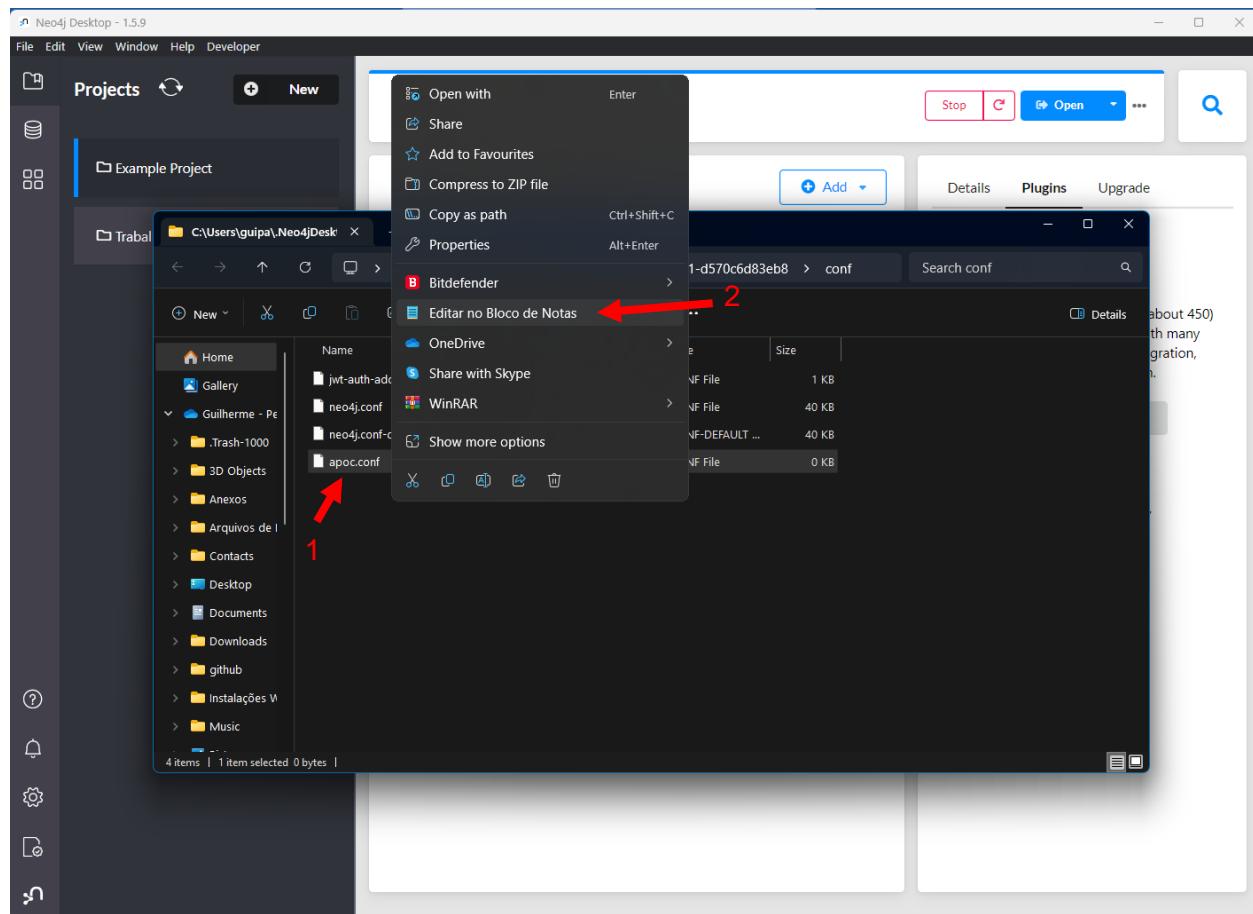
## 12. Clique em + New -&gt; Documento de texto:



13. Apague todo o nome do arquivo (incluindo o .txt que não está marcado inicialmente) e coloque como nome **apoc.conf**. Ao apertar ENTER, o sistema perguntará se você realmente quer mudar a extensão do arquivo de txt para conf. Clique em **Sim / Yes**:



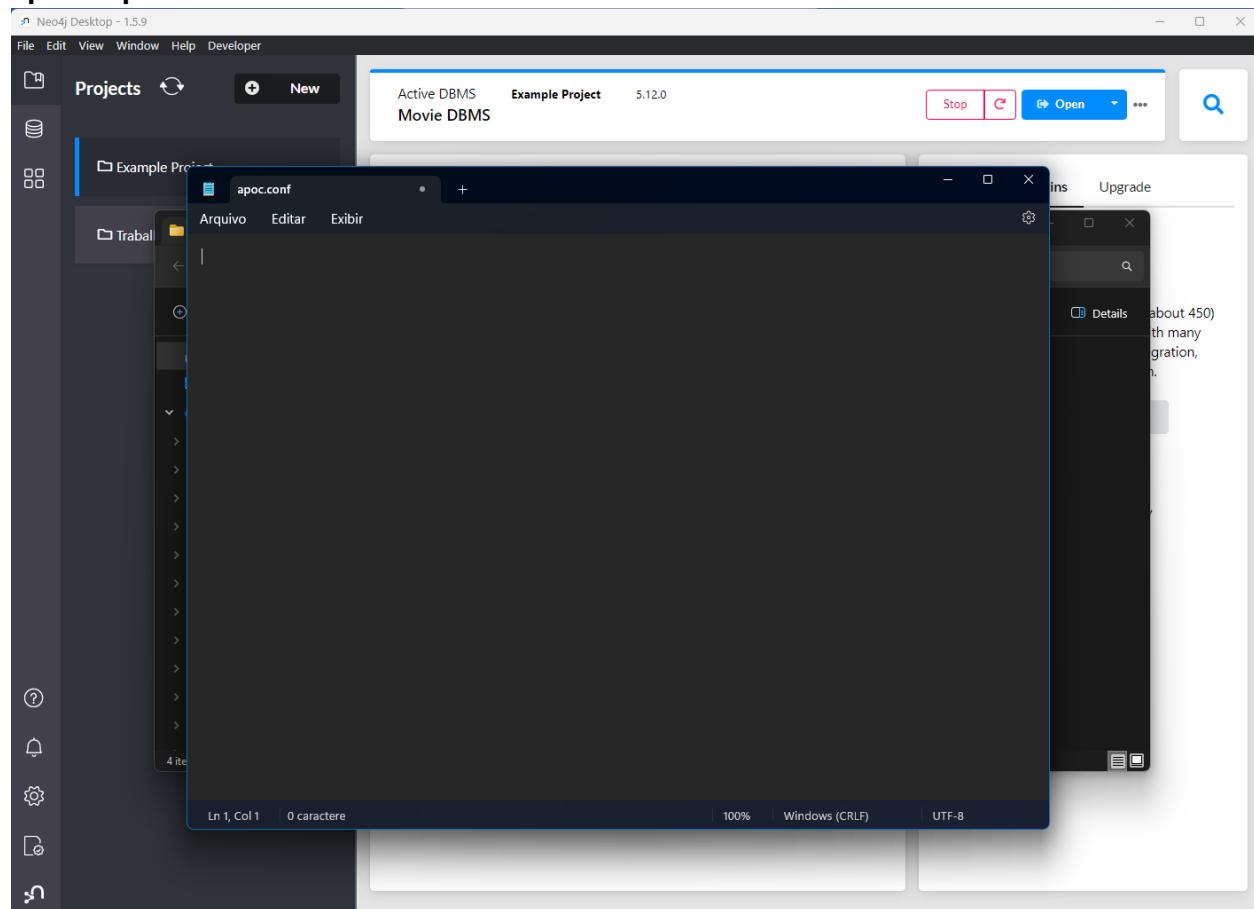
14. Clique com o botão direito do mouse no arquivo criado (apoc.conf) e escolha abrir com o bloco de notas:





15. Adicione a seguinte linha neste arquivo, salve-o e feche-o:

**apoc.import.file.enabled=true**





The screenshot shows the Neo4j Desktop interface with the title bar "Neo4j Desktop - 1.5.9". The main window displays the "Example Project" with the sub-project "Movie DBMS" selected. A context menu is open over the configuration file "apoc.conf", showing options like "Arquivo", "Editar", and "Exibir". The file content pane contains the line of code "apoc.import.file.enabled=true". The status bar at the bottom indicates "Ln 1, Col 30" and "29 caracteres".

File Edit View Window Help Developer

Projects New

Active DBMS Example Project 5.12.0

Stop Open ...

Movie DBMS

apoc.conf

Arquivo Editar Exibir

Nova guia Ctrl+N

Nova janela Ctrl+Shift+N

Abrir Ctrl+O

Salvar Ctrl+S

Salvar como Ctrl+Shift+S

Salvar tudo Ctrl+Alt+S

Configurar página

Imprimir Ctrl+P

Fechar guia Ctrl+W

Fechar janela Ctrl+Shift+W

Sair

apoc.import.file.enabled=true

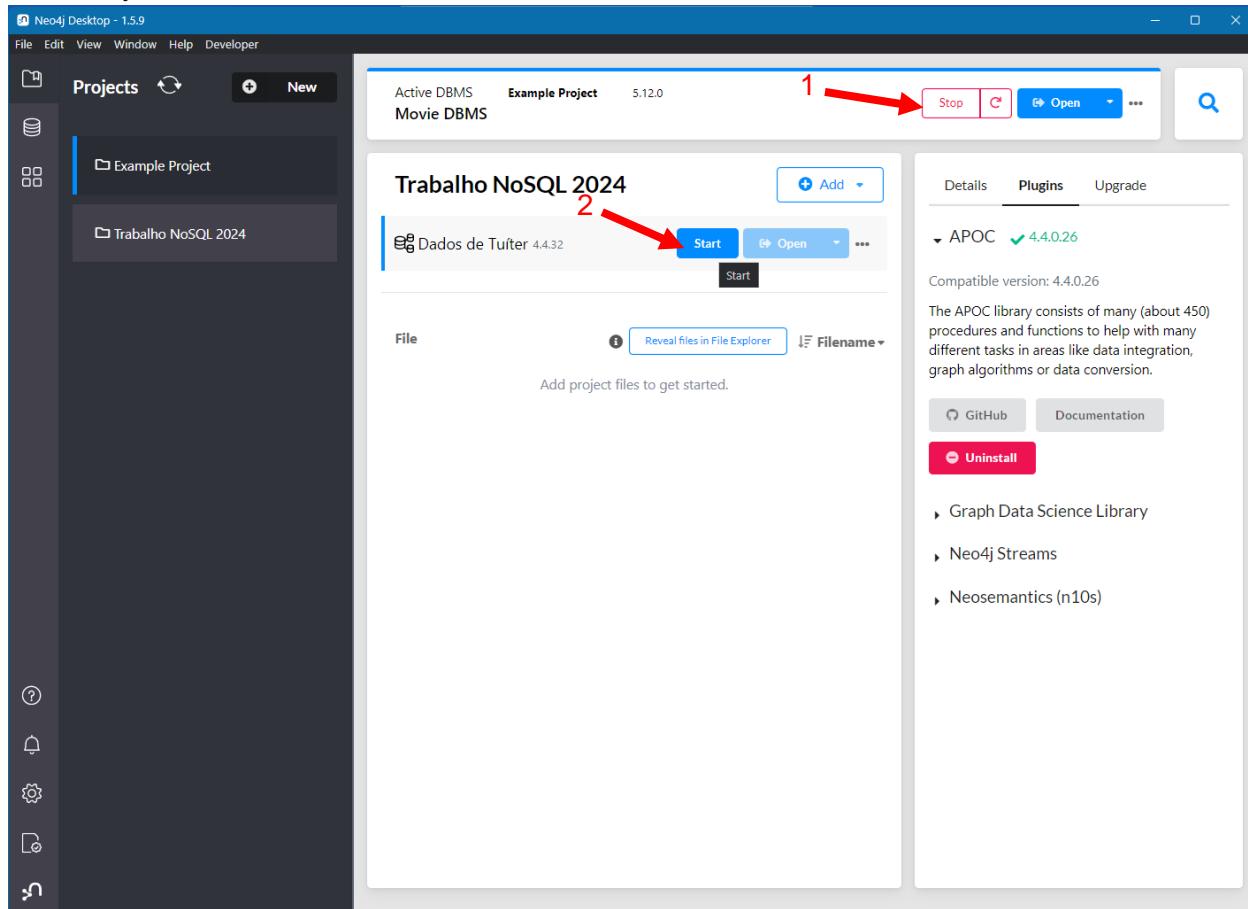
Ln 1, Col 30 29 caracteres

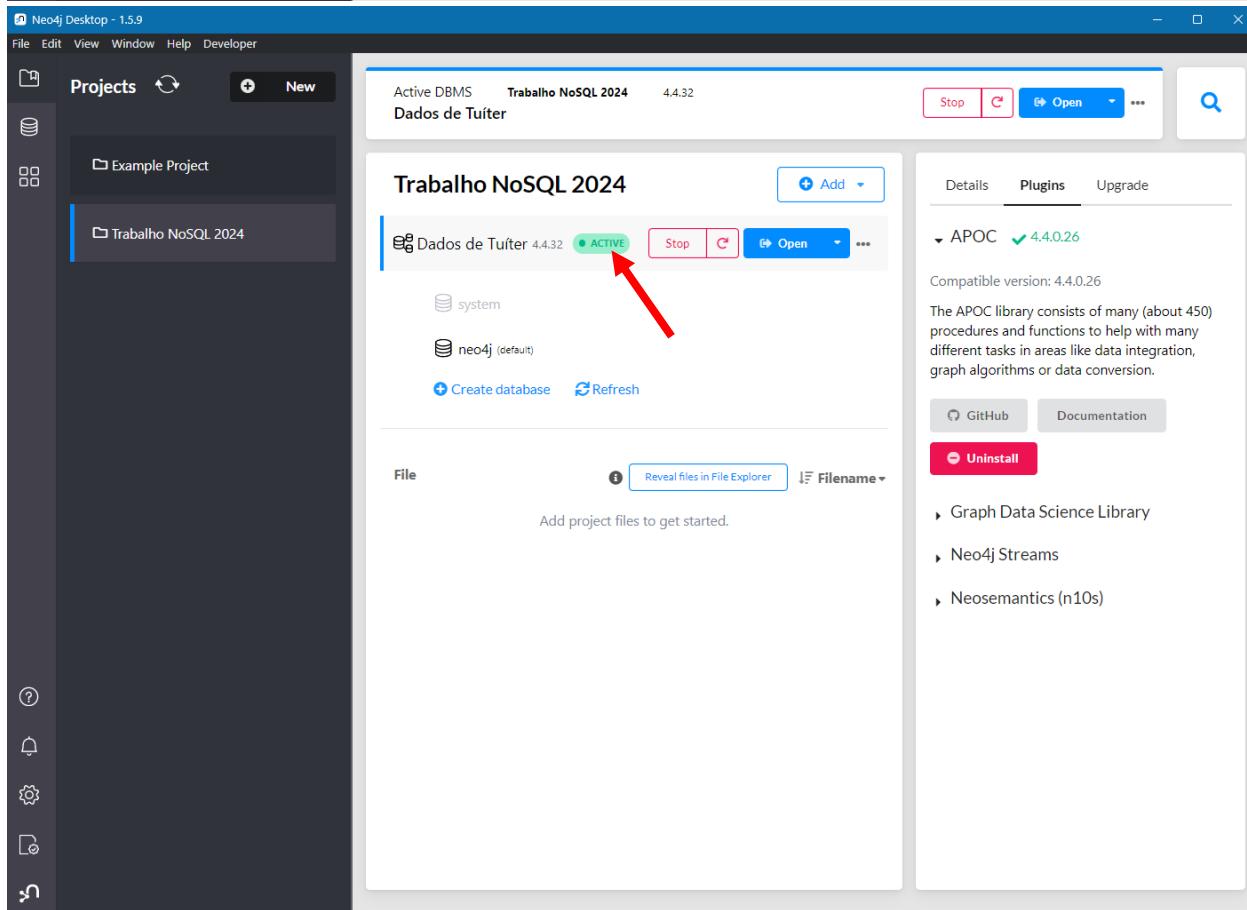
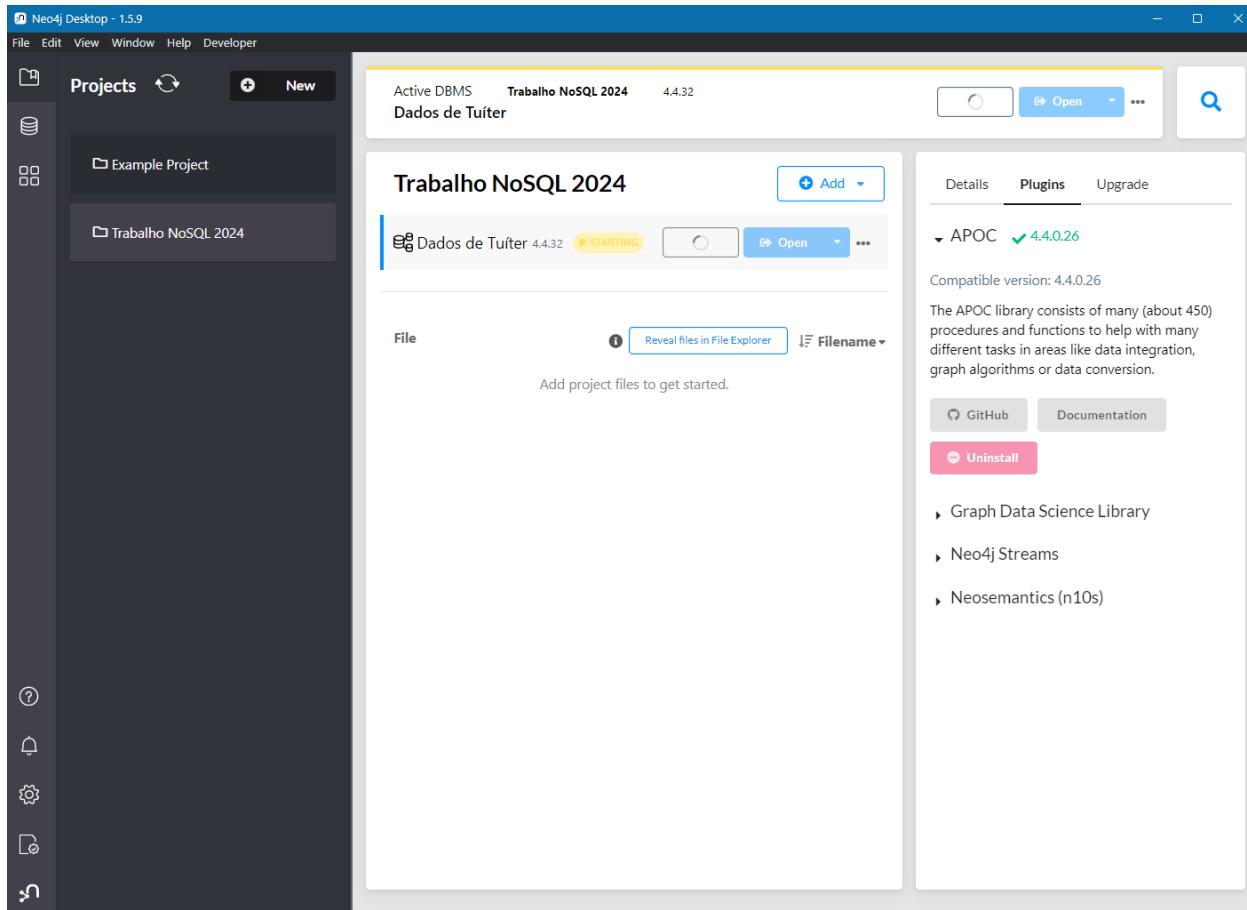
190% Windows (CRLF) UTF-8

Details about 450 with many migration, m.

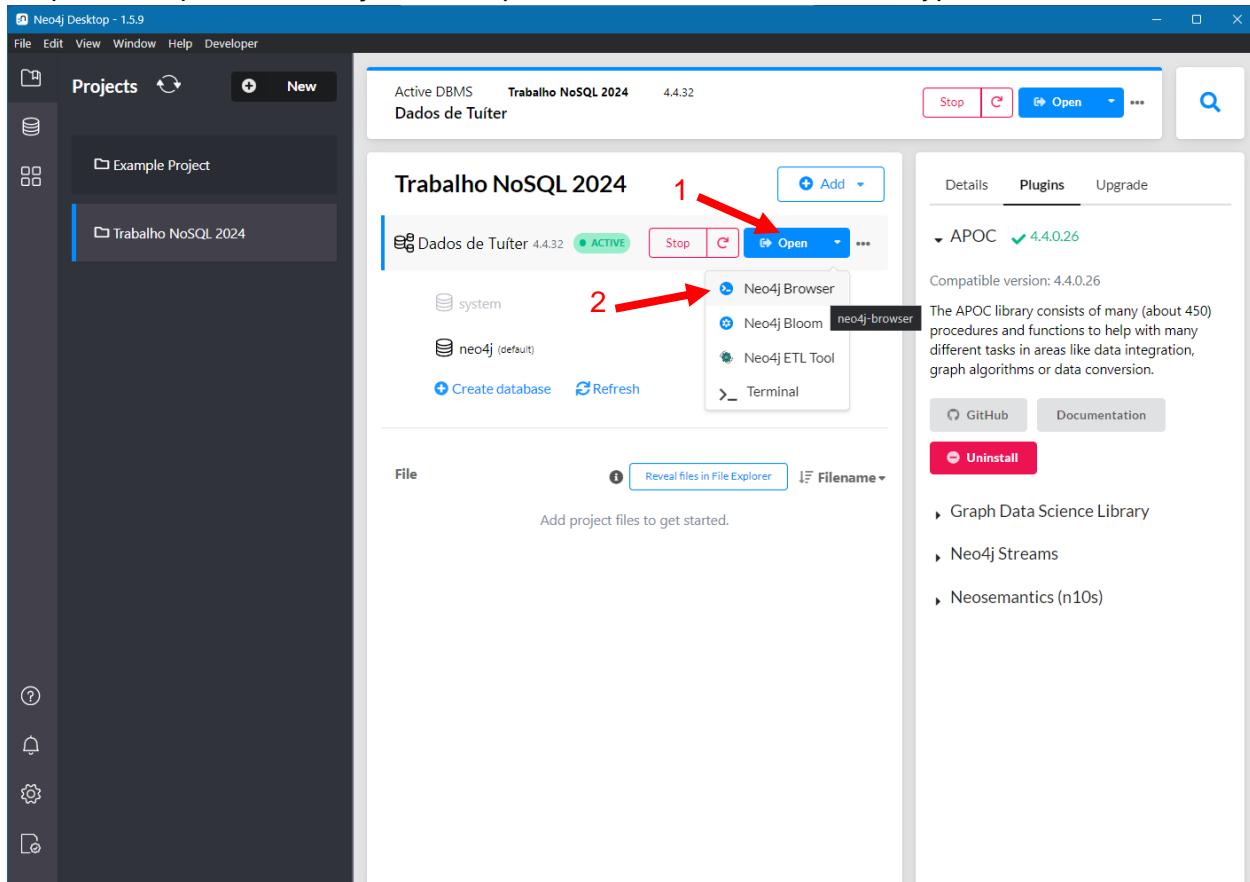
Close

### 16. No Neo4j, inicie o DBMS:





17. Clique em “Open” -> “Neo4j Browser” para abrir a tela de comandos Cypher deste DBMS:



**neo4j@bolt://localhost:7687/neo4j - Neo4j Browser**

File Edit View Window Help Developer

neo4j\$

\$ :play start

**Getting started with Neo4j Browser**  
Neo4j Browser user interface guide

**Try Neo4j with live data**  
A complete example graph that demonstrates common query patterns.  
Actors & movies in cross-referenced pop culture.

**Cypher basics**  
Intro to Graphs with Cypher  
What is a graph database?  
How can I query a graph?

Copyright © Neo4j, Inc 2002-2024

Sign up for a free Neo4j cloud instance with **neo4j aura**

\$ :server status

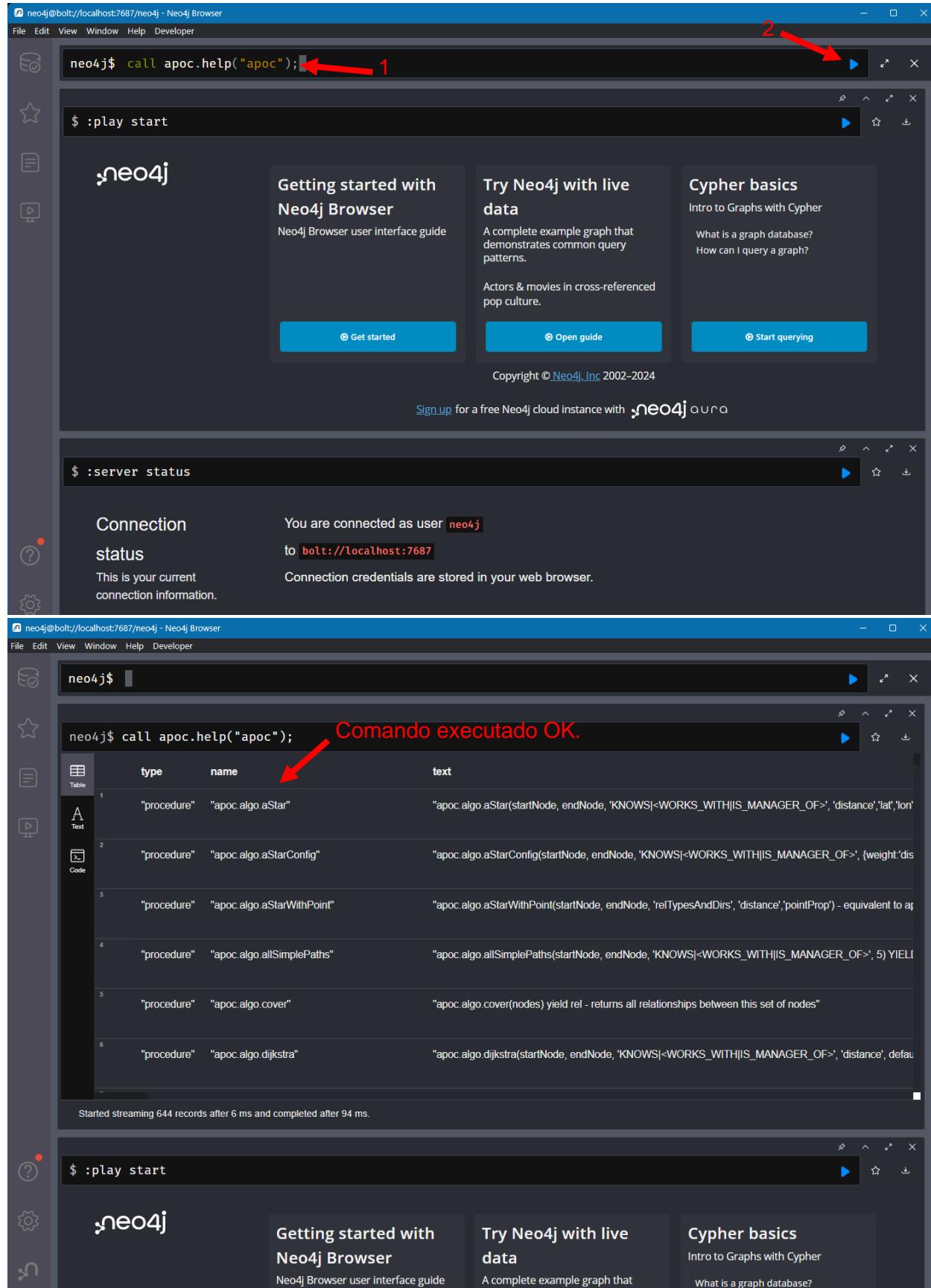
**Connection status**  
This is your current connection information.

You are connected as user **neo4j**  
to **bolt://localhost:7687**  
Connection credentials are stored in your web browser.

18. Teste se a biblioteca APOC está funcionando corretamente com o comando:

**CALL apoc.help ("apoc");**

Para executar o comando, aperte no ícone de “play” azul ou pressione “enter”.

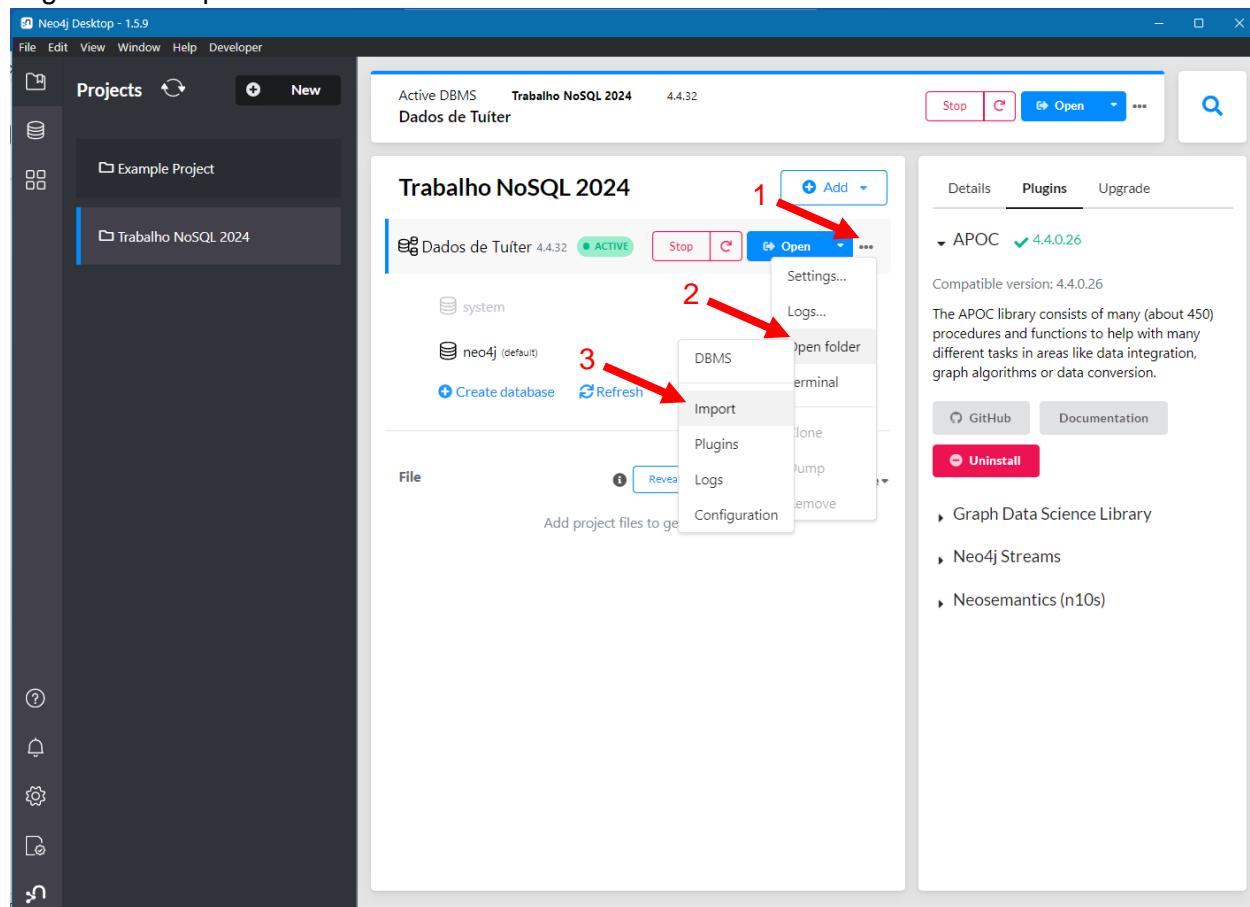


The screenshot shows two consecutive screenshots of the Neo4j Browser interface. In the first screenshot, a red arrow labeled '1' points to the command input field where 'neo4j\$ call apoc.help("apoc");' is typed. Another red arrow labeled '2' points to the blue 'play' button at the top right of the input field. In the second screenshot, the command has been executed, and the results are displayed in a table. A red arrow points to the table header with the text 'Comando executado OK.' overlaid. The table lists six procedures from the APOC library:

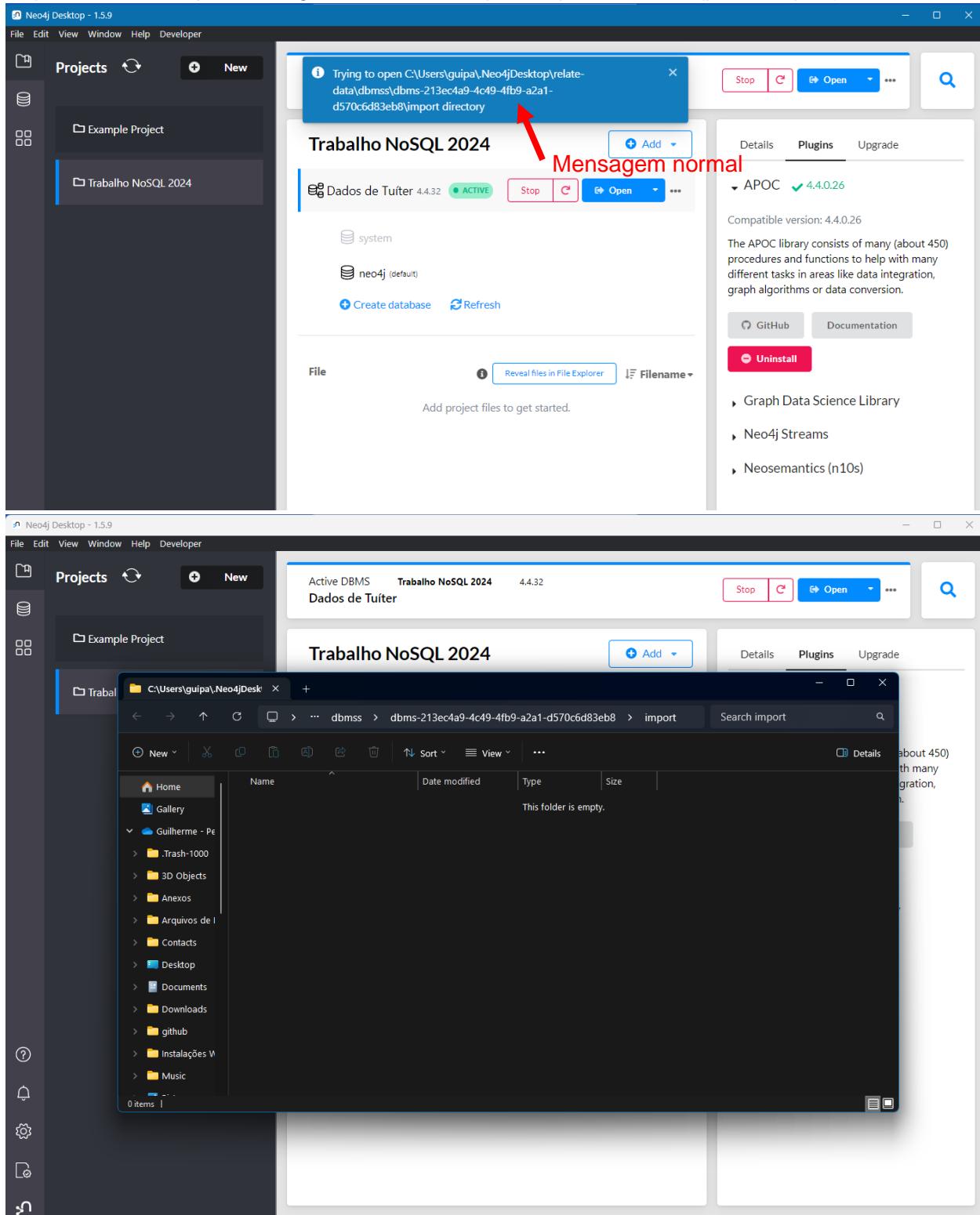
type	name	text
procedure	"apoc.algo.aStar"	"apoc.algo.aStar(startNode, endNode, 'KNOWS <WORKS_WITH IS_MANAGER_OF>', 'distance','lat','lon')
procedure	"apoc.algo.aStarConfig"	"apoc.algo.aStarConfig(startNode, endNode, 'KNOWS <WORKS_WITH IS_MANAGER_OF>', {weight:'dis')
procedure	"apoc.algo.aStarWithPoint"	"apoc.algo.aStarWithPoint(startNode, endNode, 'relTypesAndDirs', 'distance','pointProp') - equivalent to apoc.algo.aStarWithPoint(startNode, endNode, 'KNOWS <WORKS_WITH IS_MANAGER_OF>', {weight:'dis', 'pointProp'})"
procedure	"apoc.algo.allSimplePaths"	"apoc.algo.allSimplePaths(startNode, endNode, 'KNOWS <WORKS_WITH IS_MANAGER_OF>', 5) YIELD"
procedure	"apoc.algo.cover"	"apoc.algo.cover(nodes) yield rel - returns all relationships between this set of nodes"
procedure	"apoc.algo.dijkstra"	"apoc.algo.dijkstra(startNode, endNode, 'KNOWS <WORKS_WITH IS_MANAGER_OF>', 'distance', defau

At the bottom of the results table, it says 'Started streaming 644 records after 6 ms and completed after 94 ms.'

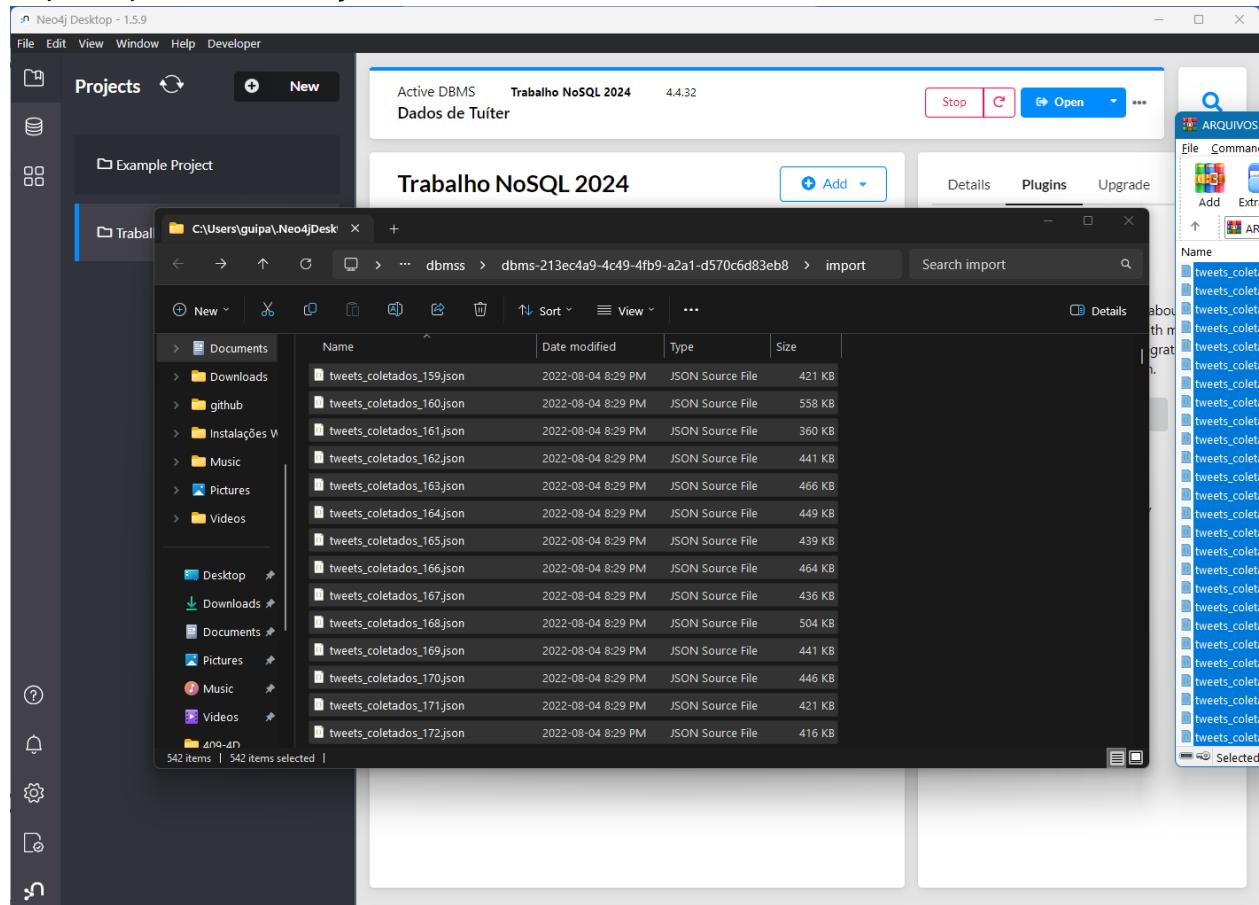
19. De volta ao Neo4j Desktop, clique nos três pontos do seu DBMS, clique em “Open folder” e em seguida em Import:



20. Uma mensagem em azul será mostrada, avisando que sua pasta foi aberta no seu gerenciador de arquivos. Vá até a janela do gerenciador de arquivos que foi aberta (pasta Import):



21. Coloque nesta pasta alguns arquivos JSON do trabalho. Para testes iniciais, sugiro colocar apenas 2 arquivos e para o trabalho final, sugiro usar apenas 5 arquivos. Neste exemplo usarei todos os arquivos para demonstração:



## TESTE DO DBMS CRIADO E FUNCIONANDO COM A BIBLIOTECA APOC

- De volta ao Neo4j Browser, vamos testar se a importação de arquivos da pasta Import está funcionando corretamente. Copie o nome de um dos arquivos que você colocou na pasta Import e faça o seguinte comando (troque a parte vermelha pelo nome do seu arquivo com a extensão .json):

**CALL apoc.load.json("[nome do seu arquivo.json]") YIELD value RETURN value;**

neo4j@bolt://localhost:7687/neo4j - Neo4j Browser

File Edit View Window Help Developer

```
neo4j$ CALL apoc.load.json("tweets_coletados_159.json") YIELD value RETURN value;
```

neo4j\$ call apoc.help("apoc");

**C:\Users\guipa\Neo4jDesk**

Name	Date modified	Type	Size
tweets_coletados_159.json	2022-08-04 8:29 PM	JSON Source File	421 KB
tweets_coletados_160.json	2022-08-04 8:29 PM	JSON Source File	558 KB
tweets_coletados_161.json	2022-08-04 8:29 PM	JSON Source File	360 KB
tweets_coletados_162.json	2022-08-04 8:29 PM	JSON Source File	441 KB
tweets_coletados_163.json	2022-08-04 8:29 PM	JSON Source File	466 KB
tweets_coletados_164.json	2022-08-04 8:29 PM	JSON Source File	449 KB
tweets_coletados_165.json	2022-08-04 8:29 PM	JSON Source File	439 KB
tweets_coletados_166.json	2022-08-04 8:29 PM	JSON Source File	464 KB
tweets_coletados_167.json	2022-08-04 8:29 PM	JSON Source File	436 KB
tweets_coletados_168.json	2022-08-04 8:29 PM	JSON Source File	504 KB
tweets_coletados_169.json	2022-08-04 8:29 PM	JSON Source File	441 KB
tweets_coletados_170.json	2022-08-04 8:29 PM	JSON Source File	446 KB
tweets_coletados_171.json	2022-08-04 8:29 PM	JSON Source File	421 KB
tweets_coletados_172.json	2022-08-04 8:29 PM	JSON Source File	416 KB

\$ :pl

neo4j

Getting started with Neo4j Browser

Try Neo4j with live data

Cypher basics

Intro to Graphs with Cypher

What is a graph database?

neo4j@bolt://localhost:7687/neo4j - Neo4j Browser

File Edit View Window Help Developer

```
neo4j$ CALL apoc.load.json("tweets_coletados_159.json") YIELD value RETURN value;
```

value

```
{
  "data": [
    {
      "text": "RT @depheliolopes: BNDES CONTRA O CORONAVÍRUS!  
ECONOMIA: Suspensão temporária de pagamentos de parcelas de financiamentos.  
#DEPHELIOLOPES #_",
      "context_annotations": [
        {
          "entity": {
            "id": "1220701888179359745",
            "name": "COVID-19"
          },
          "domain": {
            "id": "123",
            "description": "Ongoing News Stories like 'Brexit'",
            "name": "Ongoing News Story"
          }
        }
      ]
    }
  ]
}
```

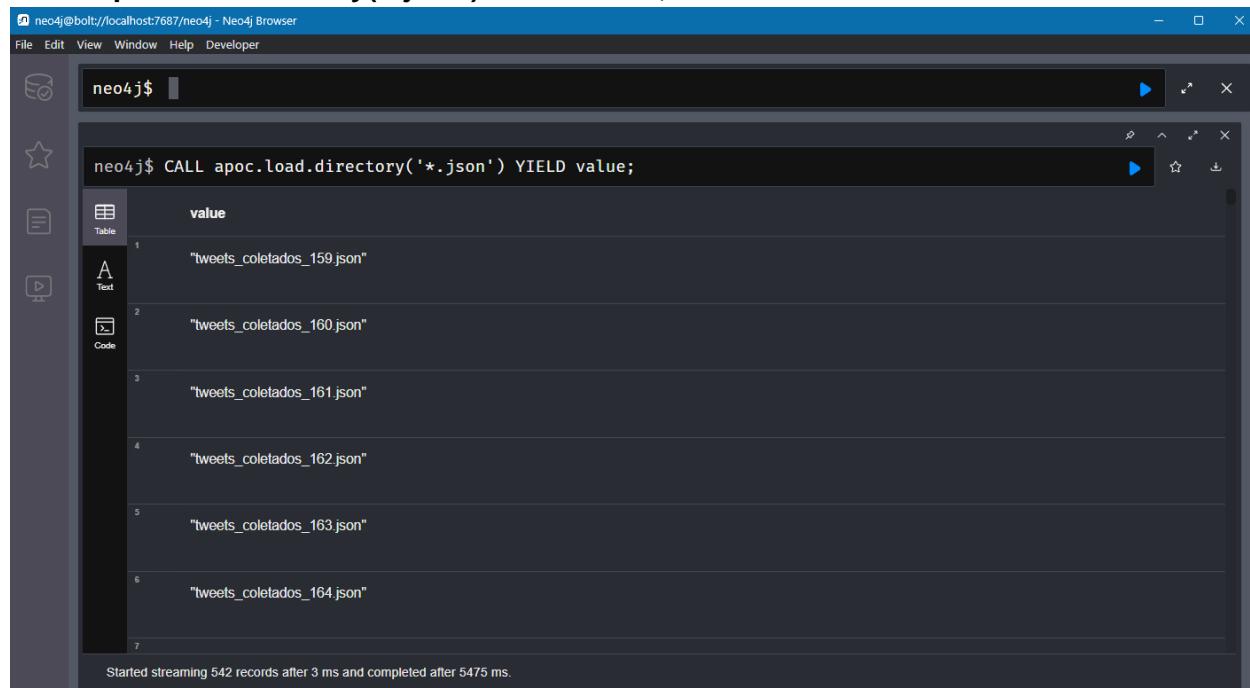
Started streaming 1 records after 3 ms and completed after 51 ms.

```
neo4j$ call apoc.help("apoc");
```

type	name	text
"procedure"	"apoc.algo.aStar"	"apoc.algo.aStar(startNode, endNode, 'KNOWS <WORKS_WITH IS_MANAGER_OF>', 'distance','lat','lon')
"procedure"	"apoc.algo.aStarConfig"	"apoc.algo.aStarConfig(startNode, endNode, 'KNOWS <WORKS_WITH IS_MANAGER_OF>', {weight:'dis

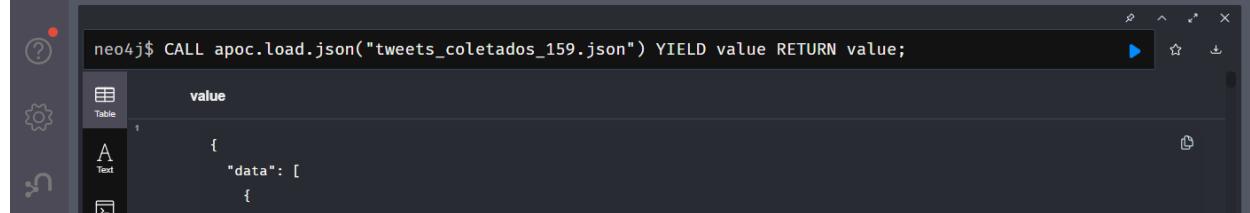
2. As configurações iniciais para o trabalho estão prontas. Vamos testar alguns comandos APOC para lermos a pasta Import completa e retornar os nomes dos arquivos presentes nela com o seguinte comando:

```
CALL apoc.load.directory('*.json') YIELD value;
```



value
"tweets_coletados_159.json"
"tweets_coletados_160.json"
"tweets_coletados_161.json"
"tweets_coletados_162.json"
"tweets_coletados_163.json"
"tweets_coletados_164.json"

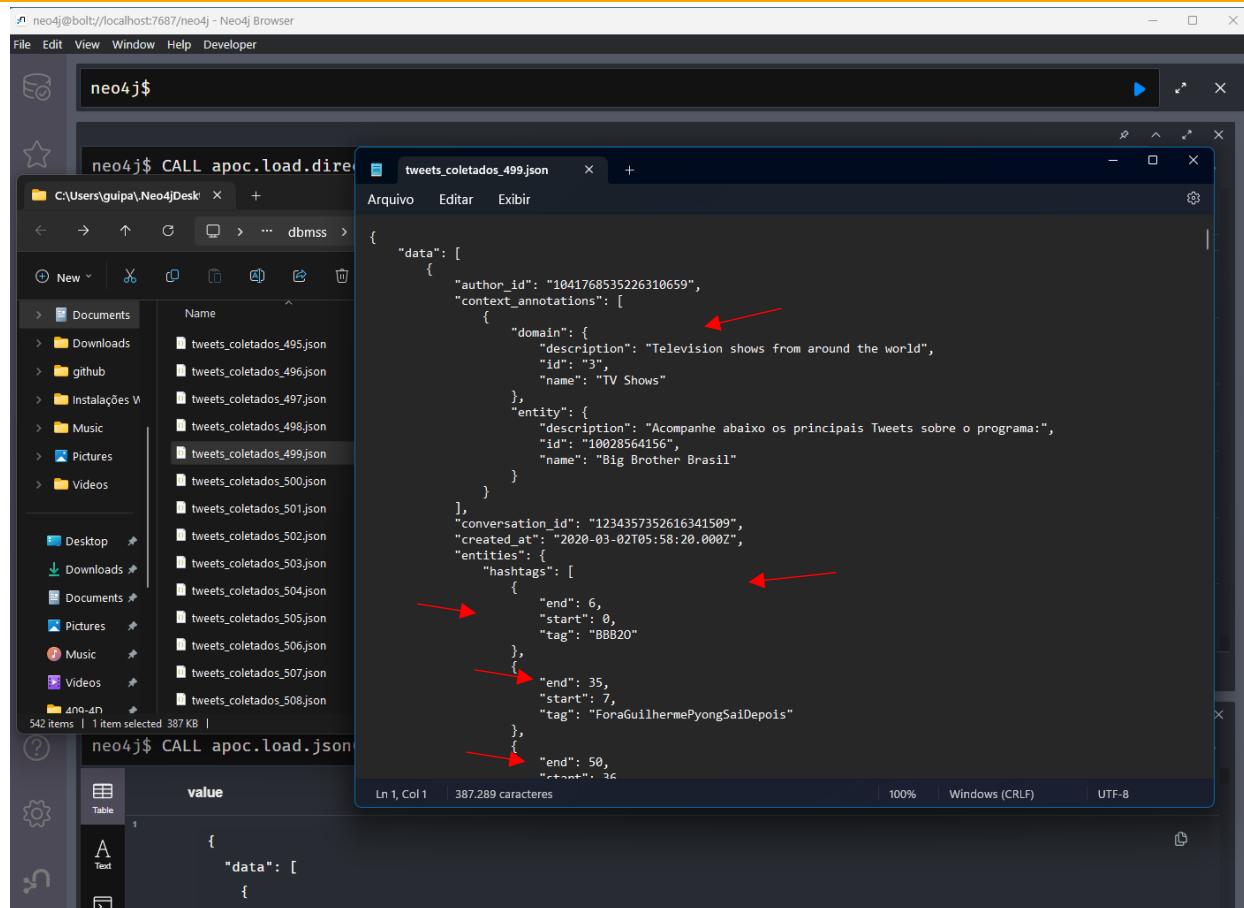
Started streaming 542 records after 3 ms and completed after 5475 ms.

value
{ "data": [ { } ] }

## EXEMPLO DE COMANDO DE CRIAÇÃO DE NÓS E RELACIONAMENTOS COM BASE NOS ARQUIVOS JSON DA PASTA IMPORT

1. Agora vamos criar nós de mensagem com base em apenas 1 arquivo da pasta Import. Usaremos o arquivo tweets\_coletados\_499.json como exemplo. Primeiro precisamos conhecer a estrutura deste JSON. Para isso, vamos abrir o arquivo no bloco de notas (botão direito em cima do arquivo e selecionar “Editar no Bloco de Notas”):



```

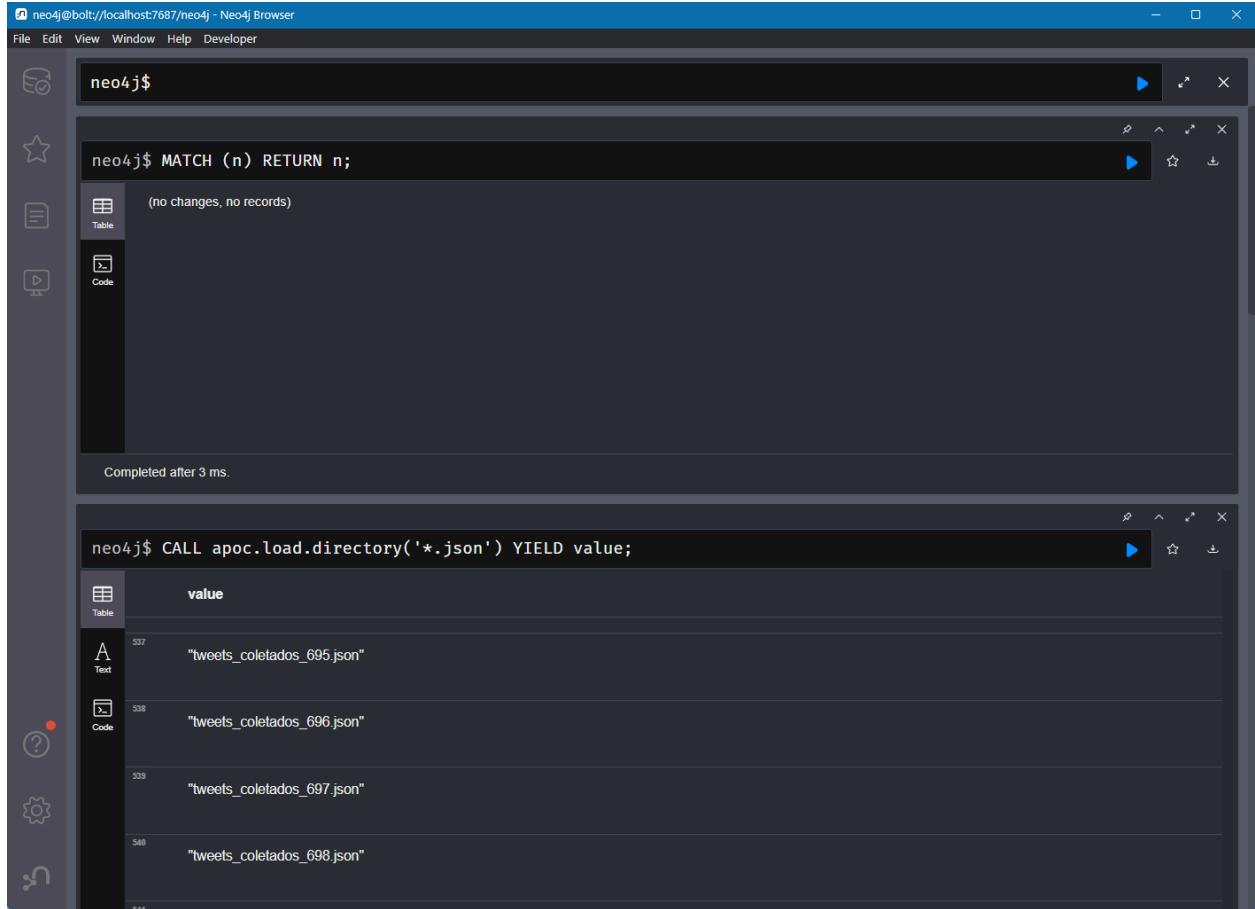
neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$ CALL apoc.load.directory('C:\Users\guipa\Desktop\Documentos\tweets_coletados_499.json')
neo4j$ CALL apoc.load.json('tweets_coletados_499.json')
    
```

The screenshot shows the Neo4j Browser interface. In the top-left, there's a file browser window showing several JSON files in the directory C:\Users\guipa\Desktop\Documentos. One file, tweets\_coletados\_499.json, is selected. In the main panel, the command `CALL apoc.load.json('tweets_coletados_499.json')` is run, and its output is displayed as a JSON object. The JSON structure includes fields like `author_id`, `context_annotations`, `domain`, `entity`, `hashtags`, and `text`. Red arrows highlight these specific fields in the JSON code.

Podemos perceber que o arquivo tem uma estrutura de lista dentro da chave “data” e que cada elemento desta lista é uma mensagem, que pode ser um tuíte original, um retuite, uma resposta para uma mensagem ou uma citação de uma mensagem. Para este trabalho vamos nos focar principalmente nos **tuítes originais**!

Também é possível encontrar informações como:

- hashtags usadas = `data.hashtags[].tag`
  - data e hora de criação da mensagem = `data.created_at`
  - id do autor da mensagem = `data.author_id`
- Para criarmos nós de mensagem, teremos que iterar na lista de mensagens (data). O mesmo vale para a criação de nós de usuário e de hashtags. Na mesma iteração será possível criar tanto o nó de mensagem, quanto o nó de usuário e os nós das hashtags usadas nesta mensagem. Veja que para hashtags, também temos uma lista na qual teremos que iterar para acharmos todas as hashtags. Nestas mesmas iterações, já podemos criar os devidos relacionamentos entre os nós, para não perdemos a informações de qual usuário postou qual mensagem e quais hashtags foram usadas.
  - Vamos criar um comando simples para criar apenas nós de mensagem, usando o comando UNWIND para iterarmos na lista data:  
Mostrando que o banco está vazio com o comando:  
**MATCH (n) RETURN n;**



The screenshot shows the Neo4j Browser interface with two main sections. The top section displays the result of the query `neo4j$ MATCH (n) RETURN n;`, which returns "(no changes, no records)" and completes after 3 ms. The bottom section displays the result of the query `neo4j$ CALL apoc.load.directory('*json') YIELD value;`. The results are shown in a table with a single column labeled "value", containing four rows of JSON file names: "tweets\_coletados\_695.json", "tweets\_coletados\_696.json", "tweets\_coletados\_697.json", and "tweets\_coletados\_698.json".

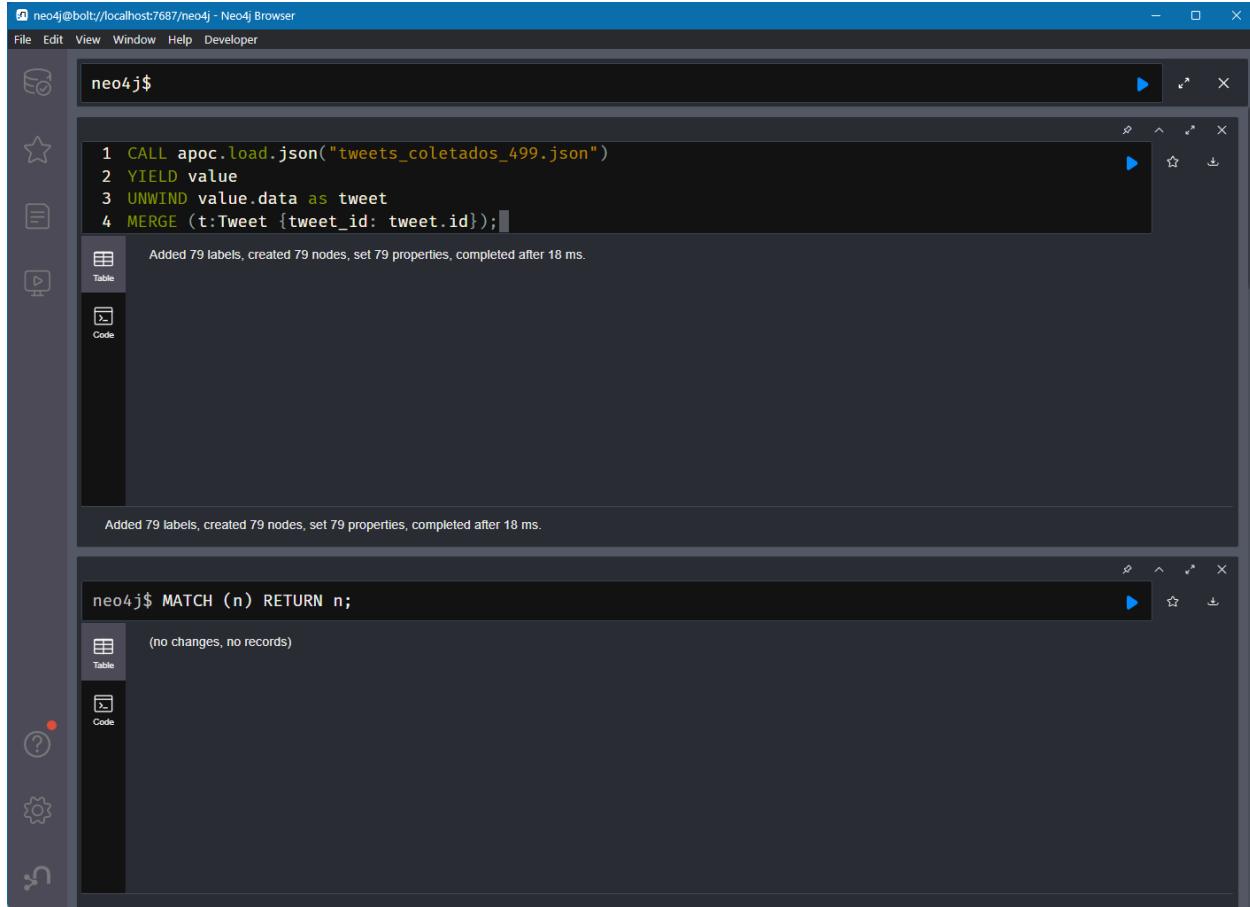
Agora sim, realizando o comando de criação simples:

**CALL apoc.load.json("tweets\_coletados\_499.json")**

**YIELD value**

**UNWIND value.data as tweet**

**MERGE (t:Tweet {tweet\_id: tweet.id});**



The screenshot shows the Neo4j Browser interface with two query panes.

**Query 1:**

```
1 CALL apoc.load.json("tweets_coletados_499.json")
2 YIELD value
3 UNWIND value.data AS tweet
4 MERGE (t:Tweet {tweet_id: tweet.id});
```

Output:

Added 79 labels, created 79 nodes, set 79 properties, completed after 18 ms.

**Query 2:**

```
neo4j$ MATCH (n) RETURN n;
```

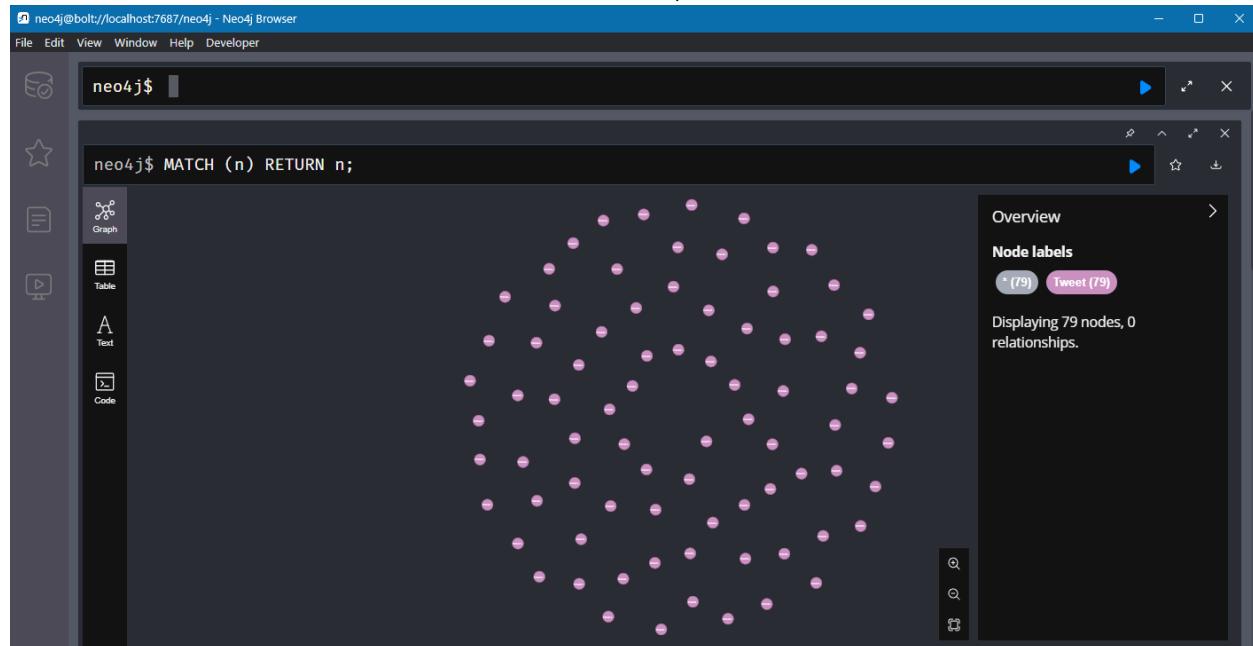
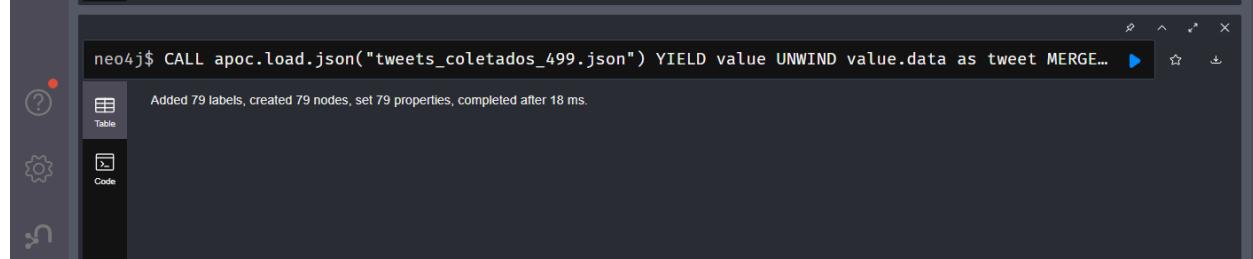
Output:

(no changes, no records)

Veja que o Neo4j retorna a quantidade de nós criados (79 nós), a quantidade de etiquetas adicionadas em nós (79 labels, um para cada nó e com o texto “Tweet”, que é o texto que colocamos depois dos dois pontos na declaração do alias dos nós “t”).

Para vermos todos os dados criados em nosso banco, podemos gerar um grafo de visualização do banco todo usando novamente o comando **MATCH (n) RETURN n** (Atenção! Este comando não deve ser usado em bancos de dados grandes pois pode travar o computador, sugerimos o uso de um limitador ou um filtro, como **MATCH (n) RETURN n LIMIT 10**; ou **MATCH (n:Tweet) WHERE**

`n.tweet_id="1234357352616341509" RETURN n;:`

4. Agora que conseguimos iterar na chave “data” do arquivo, podemos apagar o que fizemos até aqui, para não duplicar nada (apesar de que nada será duplicado pois estamos usando o comando MERGE para gerar os nós e não o CREATE – O MERGE verifica se o nó já existe e apenas adiciona o que for diferente do campo entre chaves = {tweet\_id: tweet.id}). Para apagar o banco, fazemos:

**`MATCH (n) DETACH DELETE n;`**

Mensagem de êxito informando que 79 nós foram deletados.

## QUESTÃO 01: IMPORTAÇÃO DOS ARQUIVOS JSON E CRIAÇÃO DE NÓS E ARESTAS COM BASE NOS DADOS DOS ARQUIVOS

Antes de realmente buscar dados em um banco de dados, devemos popular ele com nossos dados. Para isso, vamos criar nós e relacionamentos usando as informações constantes nos arquivos JSON de nossa pasta import, que acabamos de colocar.

Usaremos a biblioteca APOC para ler os arquivos JSON e criar nossos nós e relacionamentos com base nestes dados.

Na **parte I** da questão 01, apresente os comandos de criação do banco e na **parte II** mostre os prints de execução estes comandos (**não deverão ser mostrados nenhum print contendo grafos**).

### DICAS PARA A QUESTÃO 01:

Primeiro, vamos criar seu nó de Identificador Pessoal:

```
MERGE (ru:RU {RU: "RU-12345678"}) ;
```

(Substitua o número pelo seu RU)

Em seguida:

- Como nosso comando de criação de nós de mensagem, feito na seção anterior “EXEMPLO DE COMANDO DE CRIAÇÃO DE NÓS E RELACIONAMENTOS COM BASE NOS ARQUIVOS JSON

DA PASTA IMPORT”, criou apenas 1 atributo em cada nó, o tweet\_id com o valor existente em data.id (que é igual à tweet.id, pois renomeamos data para tweet), vamos agora criar estes mesmos nós, mas agora com mais dados (o ideal é colocarmos todos os dados que serão necessário no futuro, para não precisarmos mais ficar recriando o banco de dados). Além deles, vamos criar os nós de usuário e de hashtags. Abaixo, vamos analisar o comando que usaremos:

```

// Primeira parte do comando permanece igual. Importa o conteúdo do arquivo JSON para a variável
value
CALL apoc.load.json("tweets_coletados_499.json")
YIELD value

// Esta próxima parte fará a iteração dentro da chave “data”,
// renomeado para tweet.
UNWIND value.data AS tweet

// Aqui vamos criar os nós de mensagem como antes:
MERGE (t:Tweet {tweet_id: tweet.id})

// Aqui vamos adicionar mais atributos para nossos nós de mensagem.
// É importante termos ao menos as informações de texto, data de criação e
// ref_type (para sabermos se é tweet original ou retweet ou citação ou resposta
// para)
ON CREATE SET      t.text = tweet.text,
                    t.created_at = datetime(tweet.created_at),
                    t.conversation_id=tweet.conversation_id

// Para podermos adicionar a informação de referenced_tweet e ref_type,
// precisamos fazer uma iteração extra, pois este campo pode conter uma lista.
// Além disto, não podemos usar o UNWIND pois este campo é opcional e
// quando a mensagem é original, estes campos não existem, causando erros
// na importação (os nós não serão criados). Sendo assim, usaremos a iteração
// do FOREACH, que só é realizada quando o dado existir e não retorna erro se
// não existir. Usamos o coalesce para criar a lista vazia se não existir, para nos
// permitir adicionar elementos na lista.

FOREACH ( ref_tweet IN tweet.referenced_tweets |
            SET t.tipo_ref = coalesce(t.tipo_ref, []) + [ref_tweet.type],
            t.id_ref = coalesce(t.id_ref, []) + [ref_tweet.id]
        )

// Agora vamos criar o nó do usuário desta mensagem e fazer o relacionamento
// dele com o nó de mensagem:
MERGE (u:User {user_id:tweet.author_id})
MERGE (u)-[:TUITOU]->(t)

// Por último deixamos os nós de hashtag, pois usaremos o UNWIND para
// criarmos quantos nós forem necessários para cada mensagem (mensagens

```

```

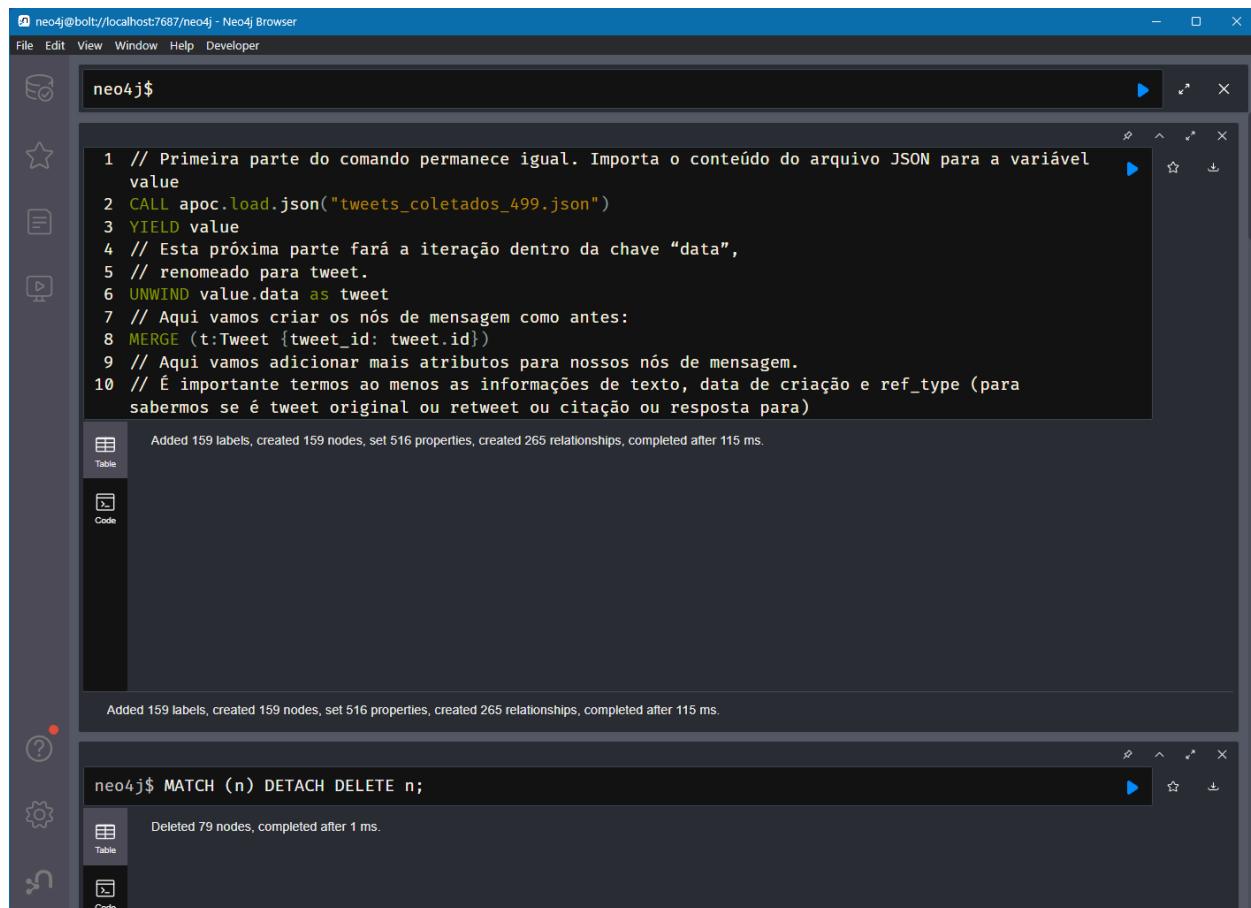
// sem hashtags serão ignoradas. Primeiro escolhemos os dados a serem
// usados, com base no comando WITH:
WITH t, tweet, tweet.entities.hashtags AS hashtags

// Agora fazemos a iteração na lista das hashtags:
UNWIND hashtags AS hashtag

// Fazemos a criação e formatação dos dados das tags, para eliminarmos
// repetições de palavras muito parecidas, como #BBB20 e #bbb20. Deixaremos
// tudo em minúsculo e sem traços ou outros elementos que não sejam letras
// ou números. Para isso usaremos o apoc.text.clean (retira acentos e
// transformas maiúsculas em minúsculas) e o apoc.text.replace com o RegEx
// [^a-zA-Z0-9] para substituir tudo o que não for número ou letra para " (vazio).
WITH t, apoc.text.replace(apoc.text.clean(hashtag.tag), '[^a-zA-Z0-9]', '')
AS cleanedHashtag

// Criação dos nós de Hashtag:
MERGE (h:Hashtag {tag: cleanedHashtag})

// Para aproveitarmos uma única leitura do arquivo JSON, vamos finalizar com a criação do
// relacionamento entre o nó de mensagem e todos os nós de hashtag criados nesta iteração:
MERGE (t)-[:POSSUI]->(h);
    
```



```

1 // Primeira parte do comando permanece igual. Importa o conteúdo do arquivo JSON para a variável
  value
2 CALL apoc.load.json("tweets_coletados_499.json")
3 YIELD value
4 // Esta próxima parte fará a iteração dentro da chave "data",
5 // renomeado para tweet.
6 UNWIND value.data as tweet
7 // Aqui vamos criar os nós de mensagem como antes:
8 MERGE (t:Tweet {tweet_id: tweet.id})
9 // Aqui vamos adicionar mais atributos para nossos nós de mensagem.
10 // É importante termos ao menos as informações de texto, data de criação e ref_type (para
    sabermos se é tweet original ou retweet ou citação ou resposta para)
    
```

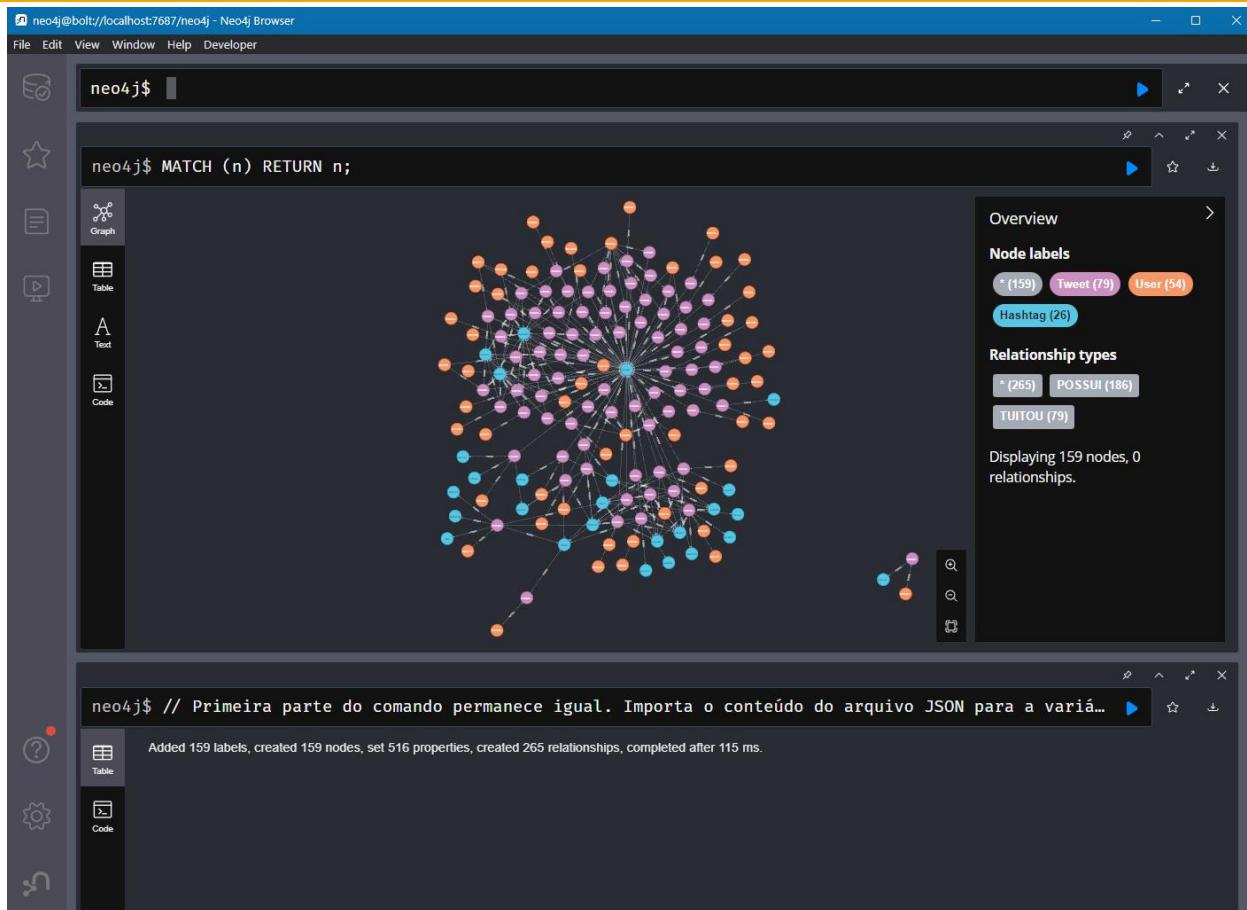
Added 159 labels, created 159 nodes, set 516 properties, created 265 relationships, completed after 115 ms.

```

neo4j$ MATCH (n) DETACH DELETE n;
    
```

Deleted 79 nodes, completed after 1 ms.

- Finalmente, veja que foram criados 159 labels (etiquetas), 159 nós, configuradas 516 propriedades (atributos) nos nós e criados 265 relacionamentos entre os nós. Para vermos o resultados, faremos MATCH (n) RETURN n (usamos este comando somente por termos menos de 300 nós no banco):



Veja que se você estiver trabalhando pelo neo4j online, será necessário incluir os relacionamentos no seu comando MATCH/RETURN para que eles sejam mostrados.

3. Para incluir neste comando todos os arquivos da pasta Import, basta incluir o comando apoc.load.directory no início, um UNWIND para iterar na lista de arquivos e um WITH para continuar os demais comandos apenas com a variável arquivo na memória, para não dar conflito de nome já existente com value (CUIDADO! Este comando poderá demorar bastante para ser completado a depender da quantidade de arquivos na sua pasta Import. Em meu caso levou 8 minutos para os 500 arquivos, resultando em 79.453 nós e labels, além de 133.272 relacionamentos):

// Buscar nomes dos arquivos na pasta Import:

```
CALL apoc.load.directory('*.json') YIELD value
// Fazer o UNWIND na lista de arquivos
UNWIND value AS arquivo
// Primeira parte do comando permanece igual. Importa o conteúdo do arquivo JSON
// para a variável value
WITH arquivo
CALL apoc.load.json(arquivo)
YIELD value
// Esta próxima parte fará a iteração dentro da chave "data",
// renomeado para tweet.
UNWIND value.data AS tweet
```

```

// Aqui vamos criar os nós de mensagem como antes:
MERGE (t:Tweet {tweet_id: tweet.id})

// Aqui vamos adicionar mais atributos para nossos nós de mensagem.
// É importante termos ao menos as informações de texto, data de criação e ref_type
// (para sabermos se é tweet original ou reweet ou citação ou resposta para)

ON CREATE SET      t.text = tweet.text,
t.created_at = datetime(tweet.created_at),
t.conversation_id=tweet.conversation_id

// Para podermos adicionar a informação de referenced_tweet e ref_type,
// precisamos fazer uma iteração extra, pois este campo pode conter uma lista.
// Além disto, não podemos usar o UNWIND pois este campo é opcional e
// quando a mensagem é original, estes campos não existem, causando erros
// na importação (os nós não serão criados). Sendo assim, usaremos a iteração
// do FOREACH, que só é realizada quando o dado existir e não retorna erro se
// não existir. Usamos o coalesce para criar a lista vazia se não existir, para nos
// permitir adicionar elementos na lista.

FOREACH (  ref_tweet IN tweet.referenced_tweets |
    SET t.tipo_ref = coalesce(t.tipo_ref, []) + [ref_tweet.type],
    t.id_ref = coalesce(t.id_ref, []) + [ref_tweet.id]
)

// Agora vamos criar o nó do usuário desta mensagem e fazer o relacionamento
// dele com o nó de mensagem:

MERGE (u:User {user_id:tweet.author_id})
MERGE (u)-[:TUITOU]->(t)

// Por último deixamos os nós de hashtag, pois usaremos o UNWIND para
// criarmos quantos nós forem necessários para cada mensagem (mensagens
// sem hashtags serão ignoradas. Primeiro escolhemos os dados a serem
// usados, com base no comando WITH:
WITH t, tweet, tweet.entities.hashtags AS hashtags

// Agora fazemos a iteração na lista das hashtags:
UNWIND hashtags AS hashtag

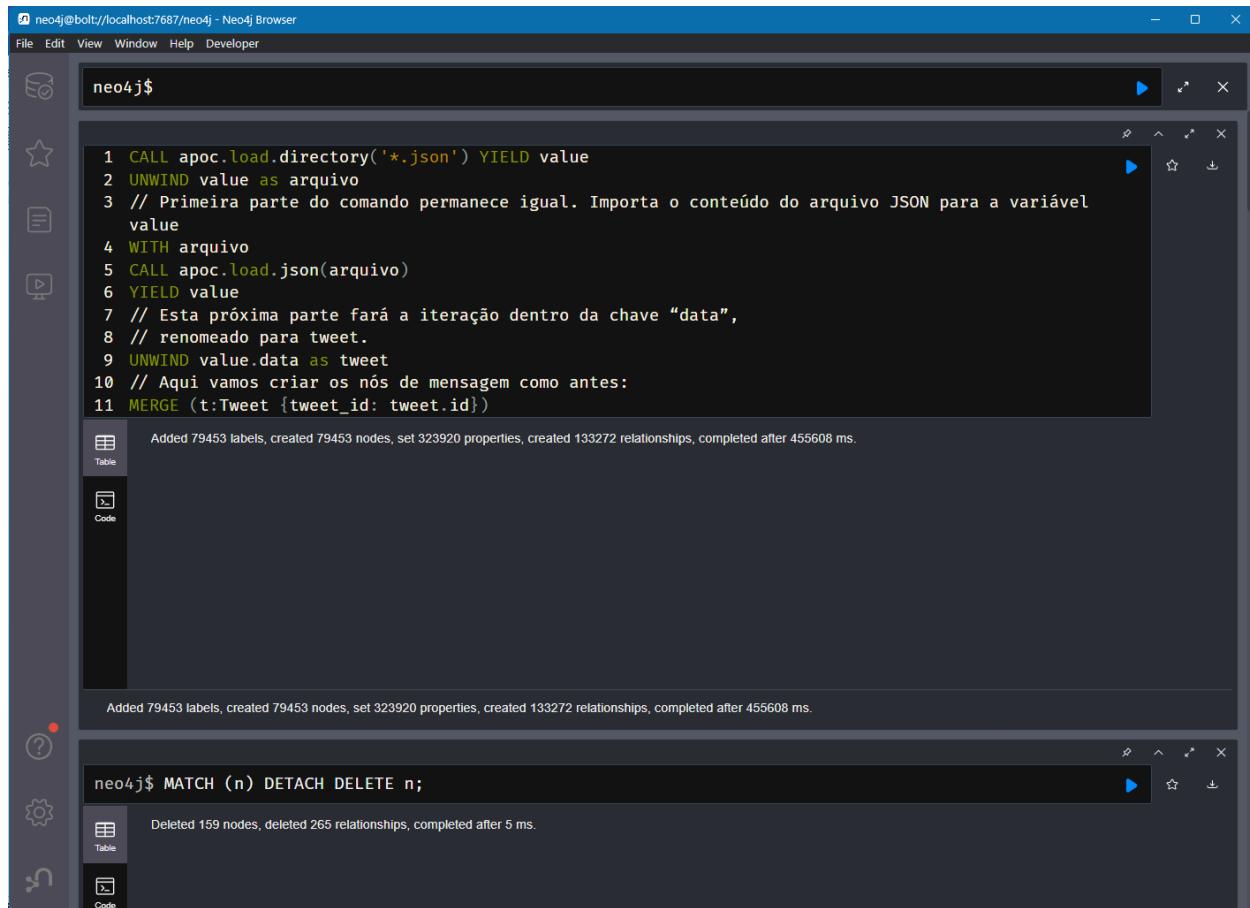
// Fazemos a criação e formatação dos dados das tags, para eliminarmos
// repetições de palavras muito parecidas, como #BBB20 e #bbb20. Deixaremos
// tudo em minúsculo e sem traços ou outros elementos que não sejam letras
// ou números. Para isso usaremos o apoc.text.clean (retira acentos e
// transformas maiúsculas em minúsculas) e o apoc.text.replace com o RegEx
// ^[a-zA-Z0-9] para substituir tudo o que não for número ou letra para " (vazio).
WITH t, apoc.text.replace(apoc.text.clean(hashtag.tag), '[^a-zA-Z0-9]', '') AS
cleanedHashtag

// Criação dos nós de Hashtag:
MERGE (h:Hashtag {tag: cleanedHashtag})

```

// Para aproveitarmos uma única leitura do arquivo JSON, vamos finalizar com a criação do relacionamento entre o nó de mensagem e todos os nós de hashtag criados nesta iteração:

```
MERGE (t)-[:POSSUI]->(h);
```



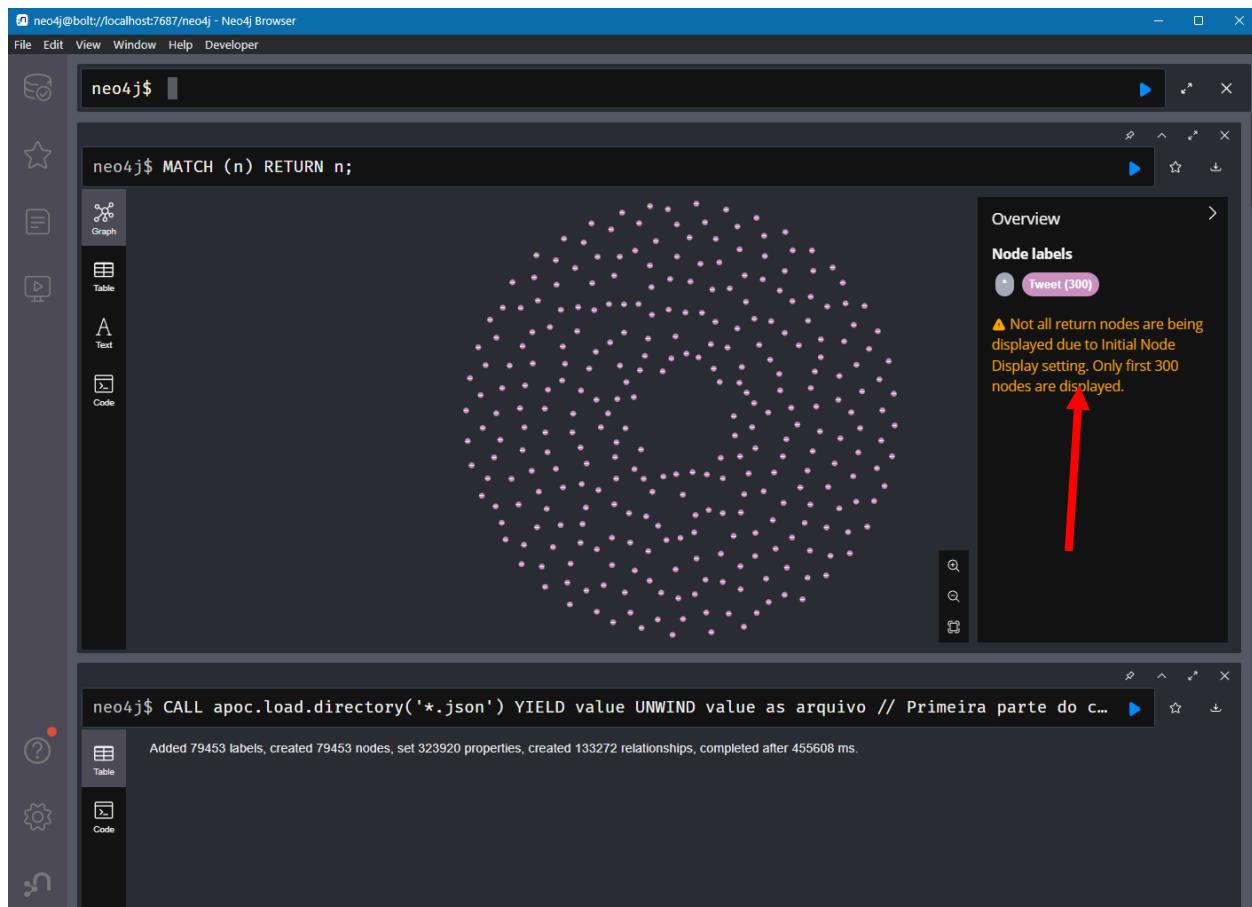
```

neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
neo4j$ 

1 CALL apoc.load.directory('*.json') YIELD value
2 UNWIND value as arquivo
3 // Primeira parte do comando permanece igual. Importa o conteúdo do arquivo JSON para a variável
4 value
5 WITH arquivo
6 CALL apoc.load.json(arquivo)
7 YIELD value
8 // Esta próxima parte fará a iteração dentro da chave "data",
9 // renomeado para tweet.
10 // Aqui vamos criar os nós de mensagem como antes:
11 MERGE (t:Tweet {tweet_id: tweet.id})
Added 79453 labels, created 79453 nodes, set 323920 properties, created 133272 relationships, completed after 455608 ms.

neo4j$ MATCH (n) DETACH DELETE n;
Deleted 159 nodes, deleted 265 relationships, completed after 5 ms.
    
```

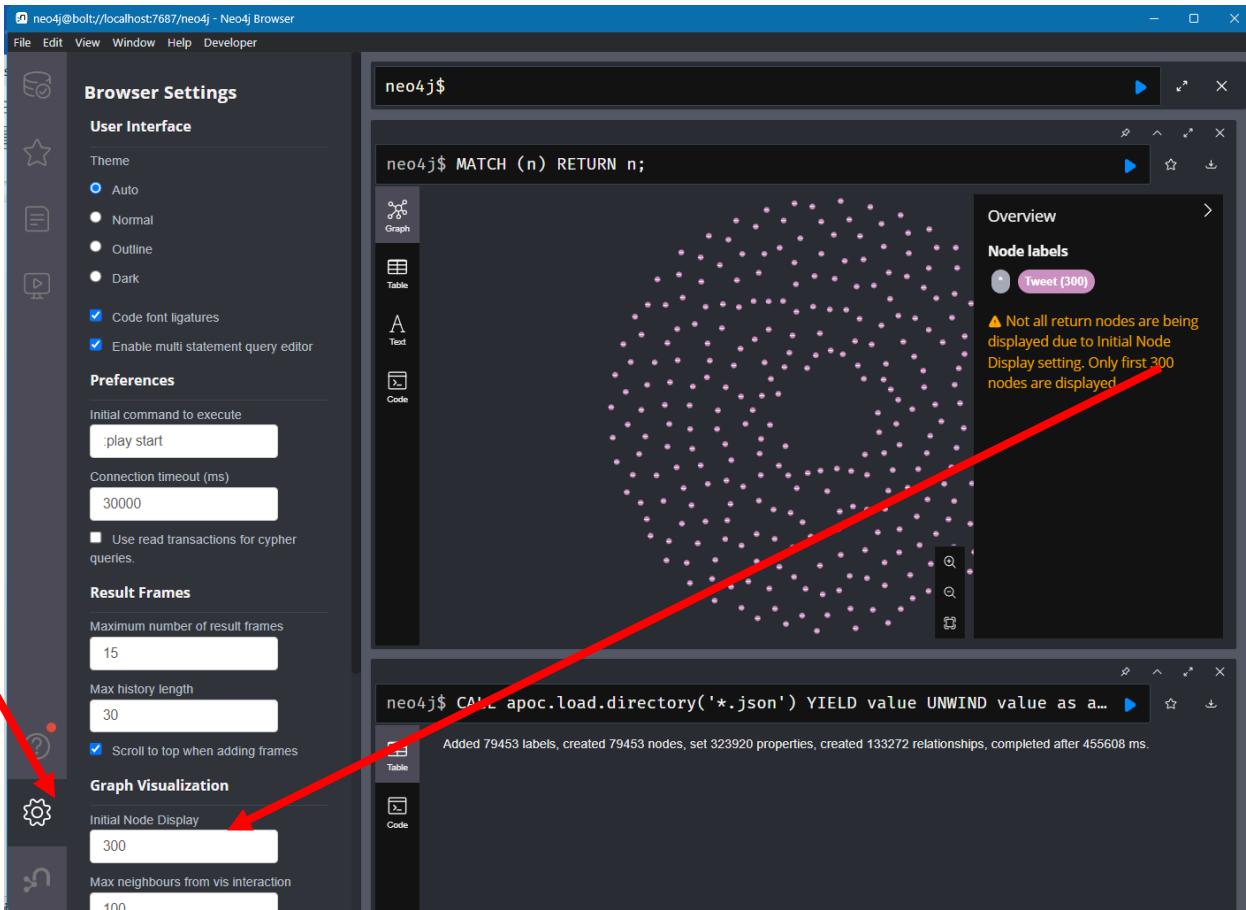
4. Agora que você tem seu banco de dados criado, basta usar o banco para encontrar as informações necessárias. Não é mais necessário usar comandos APOC e nem acessar os arquivos JSON, pois os dados já estão importados para o banco em forma de nós e arestas. Observe apenas que como o visualizador de nós possui limitação padrão de 300 nós por vez na visualização, caso você tente ver o grafo de todos os seus dados, não será possível e o resultado será algo como a imagem abaixo:



**ATENÇÃO!** Este grafo é apenas para demonstração do limite de nós e não deve ser mostrado no trabalho.

O comando MATCH (n) RETURN n; não deve ser apresentado em nenhum momento no seu trabalho.

5. Neste caso é necessário ou configurar o limite para mais nós (quanto mais nós, mais memória será consumida do seu computador, o deixando lento) ou criar comandos MATCH/RETURN com filtros específicos para analisar os dados de acordo com sua pergunta/demandra. Para aumentar o limite de 300 nós, vá em “Browser Settings” do neo4j Browser e mude o valor do campo “Graph Visualization” -> “Initial Node Display” (sugiro não passar de 3.000 nós):



**ATENÇÃO!** Este grafo é apenas para demonstração do limite de nós e não deve ser mostrado no trabalho.

O comando MATCH (n) RETURN n; não deve ser apresentado em nenhum momento no seu trabalho.

## QUESTÃO 02: DESCOBERTA DA HASHTAG PRINCIPAL

Sua missão na questão 02 será encontrar qual hashtag foi usada para gerar todos os arquivos JSON. Para isso, descubra qual delas está presente em todos os tweets (mensagens originais e não em retweets, quoted/citação nem replied\_to/resposta) coletados. Confirme sua descoberta com a geração de um grafo de visualização, usando o banco de dados já criado anteriormente, que mostre um nó central tendo esta hashtag e os demais nós como tweets que contém esta hashtag (limitar a visualização entre 10 e 20 nós). Não esqueça de mostrar seu nó de RU nesta visualização.

Sugerimos utilizar apenas 1 dia de mensagens ou menos nesta representação (não é obrigatório filtrar por data), desde que atinja o mínimo de 10 nós contando com o central, para evitar um grafo muito poluído. Caso o tempo de 1 dia seja muito grande, reduza-o até conseguir um grafo que seja visível.

Na **parte I** do seu caderno de respostas, cole todos os seus comandos utilizados para encontrar a hashtag principal. Na **parte II** do seu caderno de respostas, cole o grafo gerado pelo comando da parte I. Seu print do grafo deve estar sem zoom e mostrar o grafo e a legenda de cores dos nós à direita, como mostrado no exemplo da imagem a seguir, comprovando que apenas 1 nó de hashtag foi apresentado.

Não esqueça de colocar seu RU tanto no código quanto no grafo, antes de salvar seu arquivo em PDF.

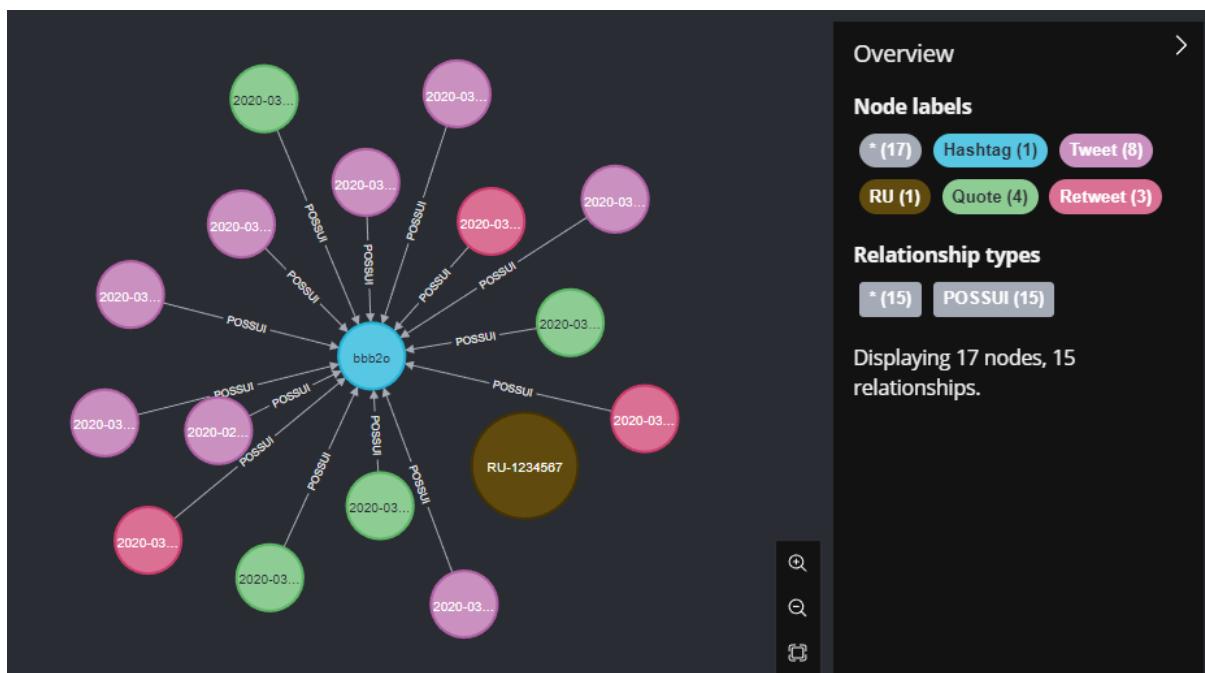
Por fim, **na parte III**, responda à pergunta: Qual foi a hashtag usada como filtro para coleta dos dados analisados? Esta hashtag só estará presente em nós do tipo Tweet e não em Retweet e a resposta deve conter apenas 1 palavra com o texto da hashtag.

Para auxílio, seguem alguns links interessantes:

- <https://neo4j.com/blog/twitter-hashtag-impact-neo4j-python-javascript/>
- <https://developer.twitter.com/en/docs/twitter-api/data-dictionary/example-payloads>

## Dicas importantes:

1. Aqui você não deve usar comandos contendo CREATE nem MERGE, bem como não deve realizar nenhuma leitura nos arquivos JSON. Esta etapa já foi realizada anteriormente para criar o banco de dados. Aqui você deve apenas usar o banco de dados criado para gerar o grafo de visualização de dados com a informação buscada.
2. Exemplo de grafo a ser gerado:



- Perceba que houve limitação de nós, que apenas 1 nó de hashtag é mostrado
- Perceba que a hashtag “bbb2o” exemplificada aqui não é resposta da questão
- Perceba que em volta do nó de hashtag temos alguns exemplos de nós de mensagem ligados à essa hashtag, incluindo Tweets, Retweets e Quotes (Citações), além do nó de RU fictício que você deve criar usando seu RU.
- Para criar este grafo de visualização, seu comando será muito parecido com estes:
  - ```
MATCH (n:NoImportante)-[r:RELACIONAMENTO_QUALQUER]-(o:OutroNo)
WITH n, COUNT(r) AS contagem
ORDER BY contagem DESC
LIMIT 1
WITH n, contagem
MATCH (n)-[r:RELACIONAMENTO_QUALQUER]-(o:OutroNo), (ru:RU)
RETURN n, o, r, contagem, ru
LIMIT 15;
```
  - ```
ou
ii. MATCH (o:OutroNo)
WITH collect(o) AS listaDeMensagens
```

```

MATCH (n:NoImportante)
WHERE ALL(o IN listaDeMensagens
          WHERE NOT isEmpty([ (n)<-[r:RELACIONAMENTO_QUALQUER]-(o) | r ]))
WITH n
MATCH (n)-[r:RELACIONAMENTO_QUALQUER]-(o:OutroNo), (ru:RU)
RETURN n, o, r, ru
LIMIT 15;
    
```

**ATENÇÃO:** Você deve criar e executar um comando Cypher em seu banco de dados para descobrir qual hashtag está presente em todas as mensagens originais (excluindo mensagens de retweet, citação e resposta). Este comando não deve fazer uso da biblioteca APOC. Caso seu comando tente retornar mais de 300 nós, a configuração padrão do Neo4j Browser impedirá a exibição de mais de 300 nós.

### Seu comando não pode ser MATCH (n) RETURN n;.

Por fim, na **parte I** cole seu(s) comando(s), na **parte II** cole o grafo gerado e inclua uma legenda explicativa sobre o grafo e como interpretá-lo e na **parte III** responda à pergunta: **Qual foi a hashtag usada como filtro para coleta dos dados analisados (esta hashtag deverá estar presente em todas as mensagens originais e sua análise deve desconsiderar nós de mensagens do tipo retweeted)?**

## QUESTÃO 03: ANÁLISE DOS DADOS SEGUNDO VIÉS A SUA ESCOLHA

Agora que você já sabe como criar um grafo de visualização a partir dos dados no seu banco de dados, o maior desafio é conseguir extrair informações úteis de grandes quantidades de dados que podem parecer desconexos.

Sua tarefa aqui é analisar os dados do banco e definir qual informação você gostaria de obter e que tenha potencial de gerar um grafo com mais de 10 nós interligados entre si e menos de 200.

Com base em sua escolha, você pode optar (não obrigatório) por definir um período para realizar sua análise que seja grande o suficiente para apresentar os resultados que você deseja, porém que não criem um grafo ilegível.

A sugestão é criar um grafo com mais de 10 nós e menos de 200 com base em uma busca nos dados do seu banco à procura de alguma informação que você ache interessante a destacar. Algumas sugestões de questões a serem respondidas são (Algumas destas opções precisam ter a inclusão de dados nos atributos dos nós com base nos dados dos arquivos JSON e devem ser incluídos na criação do seu banco e não após):

1. Qual o usuário que mais movimentou a rede de acordo com seus dados? (necessário criar nós de usuários e seus relacionamentos com os nós de mensagem)
2. Qual o equipamento mais usado para postar mensagens? (necessário criação de nós de equipamento e relacionamentos com as mensagens)
3. Quais as 3 hashtags menos usadas?
4. Qual o período (granularidade de hora) com maior movimentação da rede? (Necessário adicionar atributo com dado de created\_at)
5. Qual o dispositivo mais usado para tuítar? (depende de seu banco de dados já possuir os nós de equipamentos usados, criados na questão 01).
6. Qual a Hashtag que menos foi usada?
7. Quais os usuários mais citados? (depende dos seus nós de mensagem possuírem esta informação ou de relacionamentos terem sido criados para este fim).

Para realizar a consulta ao banco, use comandos MATCH/RETURN parecidos com o usado na questão 01. Aqui você não deve realizar nenhum comando contendo leitura dos arquivos JSON nem uso da biblioteca APOC, mas pode utilizar MERGE e CREATE caso ache necessário incluir nós e relacionamentos baseados nos dados já existentes no seu banco e não nos arquivos JSON.

Então, crie a consulta ao banco de dados que apresente os dados que você escolheu analisar e gere o grafo de sua consulta para colocação no seu caderno de respostas.

**ATENÇÃO!**

**Seu comando não pode ser MATCH (n) RETURN n;.**

Por fim, na **parte I** cole seu(s) comando(s)/query(es) com legendas explicando o funcionamento do seu comando, na **parte II** cole o grafo e inclua uma legenda sobre ele e como interpretá-lo e na **parte III** **explique qual foi a análise realizada, incluindo sua linha de raciocínio para criação da query Cypher e qual era sua expectativa de resultado antes da análise, comparando-a com o resultado realmente obtido após execução do comando.**

# RESPOSTAS AS DÚVIDAS MAIS FREQUÊNTES

## 1. Como faço para coletar meus próprios tweets?

R: Infelizmente a coleta de Tweets deixou de ser gratuita para pesquisadores após aquisição e alteração do nome da rede social de Twitter para X. No momento existem apenas coletas de mensagens em tempo real com poucas opções e baixo fluxo de dados ou contas corporativas com custos mensais superiores a 5 mil dólares para realizar o mesmo tipo de coleta.

## 2. Estou terminando o curso, tem como fazer um questionário para atividade prática?

R: Não.

## 3. Eu não possuo máquina para realizar esta atividade. Como devo proceder?

R: Você poderá usar um computador em seu polo. Nossas atividades são pensadas para execução em computador disponível nos polos, porém você pode optar por usar sua própria máquina.

## 4. Professor, eu fiz o download dos arquivos e o JSON veio dividido em um monte de arquivos pequenos. O que eu faço?

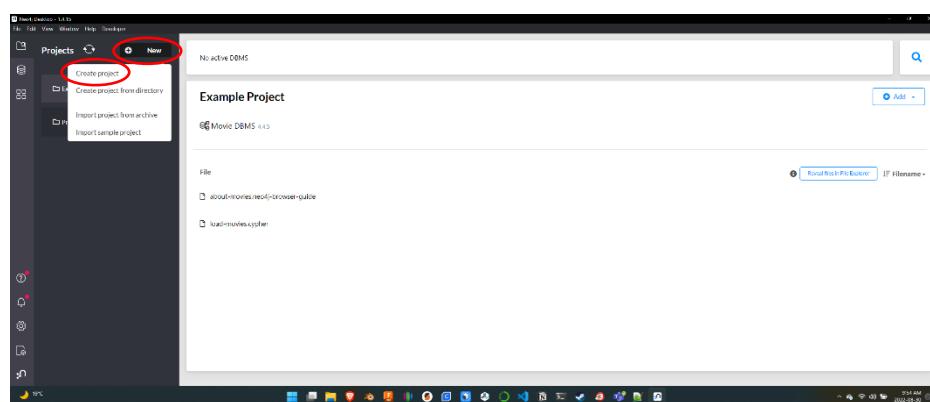
R: Conhecer o neo4j e as ferramentas para manipulação de dados JSON é de suma importância para o mercado de trabalho nesta área. Pesquise na documentação do Neo4j para todas as informações necessárias para esta importação (<https://neo4j.com/docs/>).

## 5. Professor, não estou conseguindo configurar o Neo4j para importar arquivos JSON. Só aparece erro de falta da biblioteca APOC. Como eu começo?

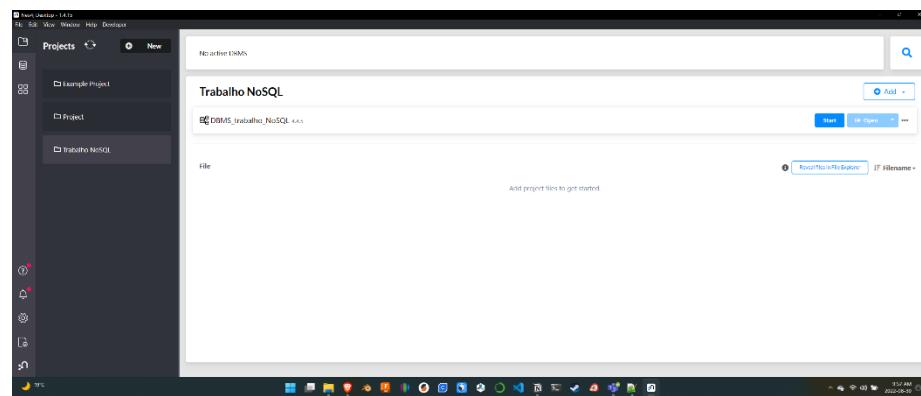
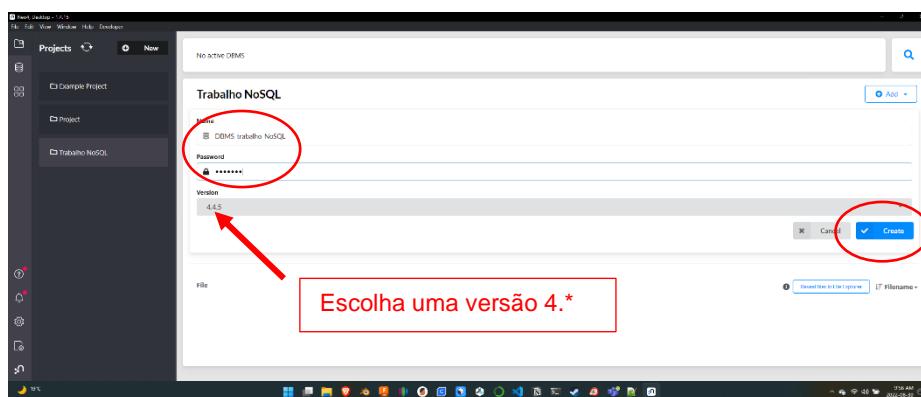
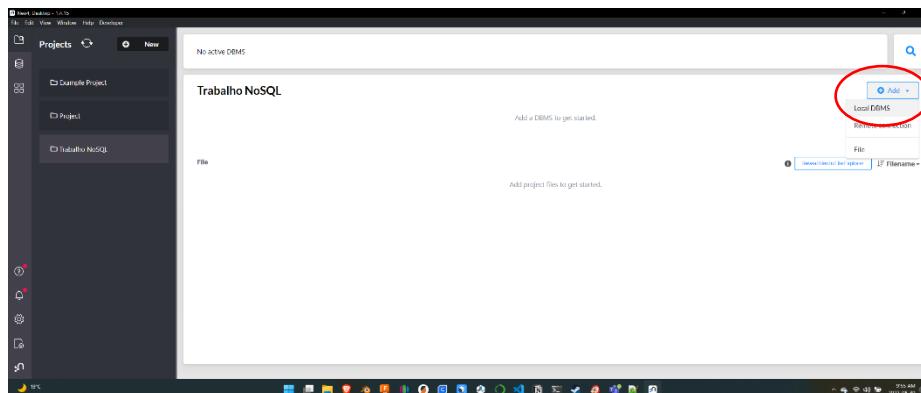
R: Para começar o trabalho, segue alguns passos que poderão agilizar o início:

### 1- Configurações iniciais:

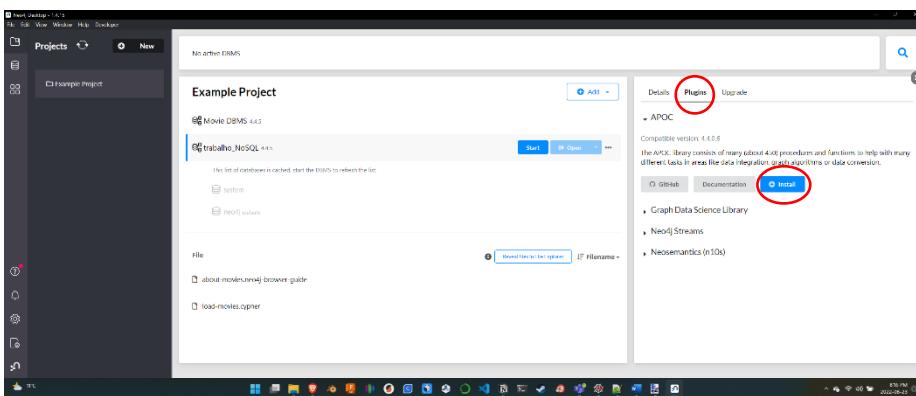
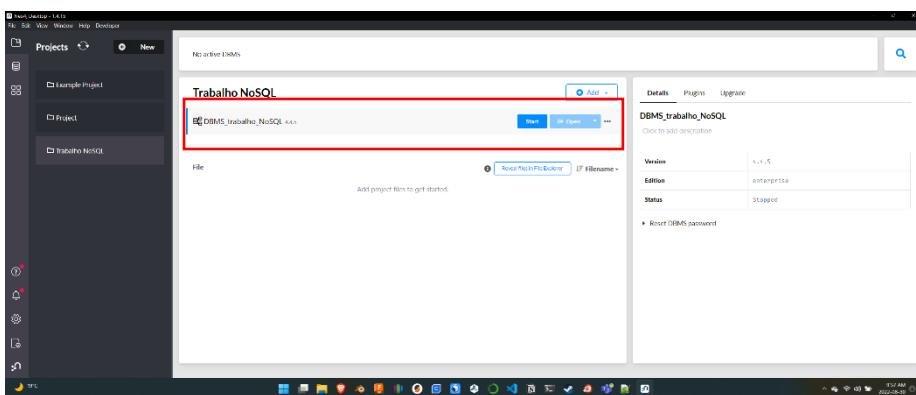
- 1.1 - Instale o Neo4j e execute-o.
- 1.2 - Crie um projeto neo4j



## 1.3 - Crie um banco de dados local dentro do novo projeto

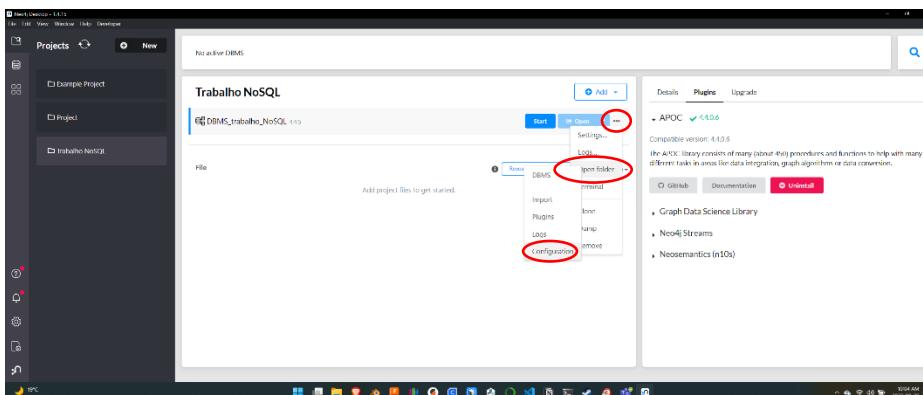


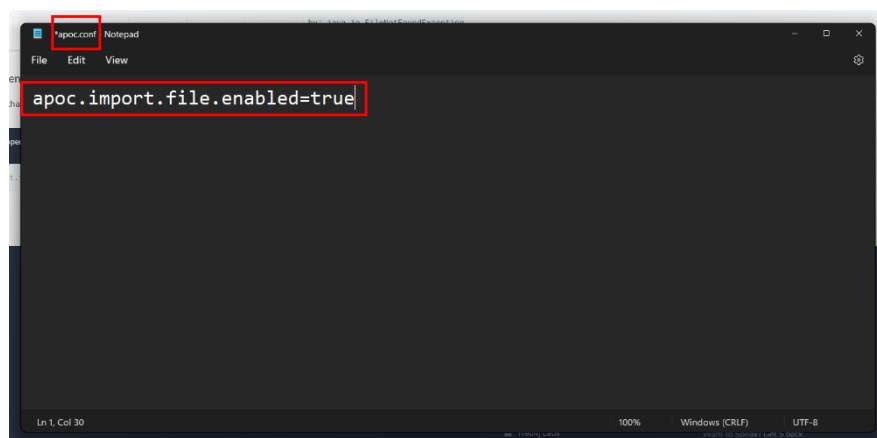
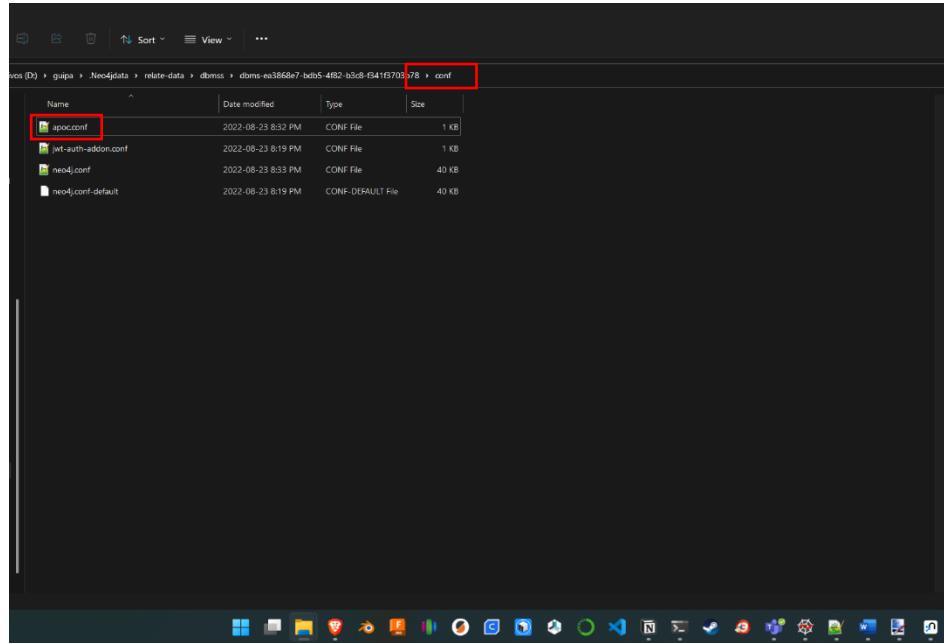
1.4 - Clique sobre o nome do banco e instale a biblioteca APOC (menu lateral direito) - Ver mais em <https://neo4j.com/labs/apoc/4.4/installation/>



1.5 - Abra a pasta de configuração do seu novo DBMS (clique nos 3 pontos ao lado do nome do seu banco → Open File → Configure), adicione um novo arquivo de texto e renomeie para apoc.config. Inclua nele a linha: (veja mais em: <https://neo4j.com/labs/apoc/4.4/config/>)

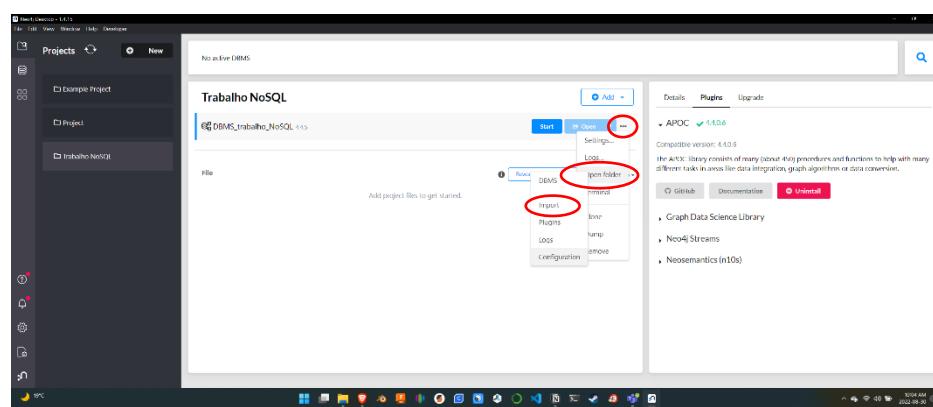
**apoc.import.file.enabled=true**





1.6 - Inicie seu DBMS criado.

1.7 - Abra a pasta "Import" do seu DBMS e crie a pasta jsonFolder, para facilitar a importação dos arquivos em lote



1.8 – Coloque um dos arquivos json do trabalho na pasta "Import" (não coloque na jsonFolder ainda) para testes.

**DICAS:**

Para este trabalho você deverá utilizar a biblioteca APOC do Neo4j. Não é necessário criação de nenhum script em linguagem de programação, apenas comandos Cypher (queries).

Não importa quantos arquivos json existam. Você só precisará de dois comandos principais, uma boa dose de lógica e bons conhecimentos dos comandos CYPHER.

Comece fazendo o exemplo de uso desta página:

<https://neo4j.com/labs/apoc/4.3/overview/apoc.load/apoc.load.json/>

Depois faça o exemplo de uso desta página:

<https://neo4j.com/labs/apoc/4.3/overview/apoc.load/apoc.load.directory.async.add/>

ou desta:

<https://neo4j.com/labs/apoc/4.3/overview/apoc.load/apoc.load.directory/>

Quando conseguir rodar os dois, faça uma tentativa com um dos arquivos .json dos dados do trabalho.

Explicação:

O comando apoc.load.json() carrega os dados de um arquivo json que esteja na pasta Import do seu DBMS e cria uma variável com eles. Assim é possível criar nós com utilizando estes dados como propriedades dos nós, que é o que queremos.

O comando apoc.load.directory.async.add() cria uma tarefa automática que será executada sempre que um novo arquivo com a extensão que você determinar for colocado em uma pasta que você definir (por exemplo, na pasta jsonFolder criado no passo 1.7 anterior).

O comando apoc.load.directory() permite criar uma lista com todos os arquivos de um determinado tipo de dentro de uma pasta e suas subpastas. Com esta lista é possível iterar nela com os nomes e caminhos de sistema para dentro do comando apoc.load.json().

Você se deparará com muitas configurações a serem feitas. Todas elas são descritas nos site mencionados anteriormente.

Crie todos os nós e relacionamentos no mesmo comando de importação do arquivo JSON, para aproveitar a leitura dos dados (no mesmo tweet tem o autor, as hashtags e a mensagem, que podem ser criados e interligados sem precisar criar filtros para isso).

Crie nós com todas as propriedades que você achar pertinente ou que poderá utilizar nas análises, como incluir data e hora de criação da mensagem, incluir id do autor, incluir todas as hashtags, etc.

Cuidado ao tentar importar todos os arquivos. Provavelmente você não precisará de todos eles para fazer o trabalho. Cabe a você analisar os dados e tomar esta decisão.

O comando UNWIND permitirá que você acesse níveis mais internos dos dados JSON, como hashtags.tag ou outros dados agrupados em listas.

Troque seu CREATE pelo MERGE. Ele criará um nó caso a propriedade com determinado valor não exista ou se encontrar um MATCH, não duplicará o nó, apenas adicionará os dados extras.

Não use o UNWIND com variáveis que não sejam do próprio arquivo. Isto só vai aumentar a complexidade e tempo de execução.

Sugestão de comando tudo numa mesma célula/de uma vez só (Cuidado! Deixe apenas 3 ou 4 arquivos na pasta para testes, pois o carregamento de tudo pode demorar até 2 horas, dependendo do computador. Só execute com tudo quando estiver certo de que é isto que você quer):

```

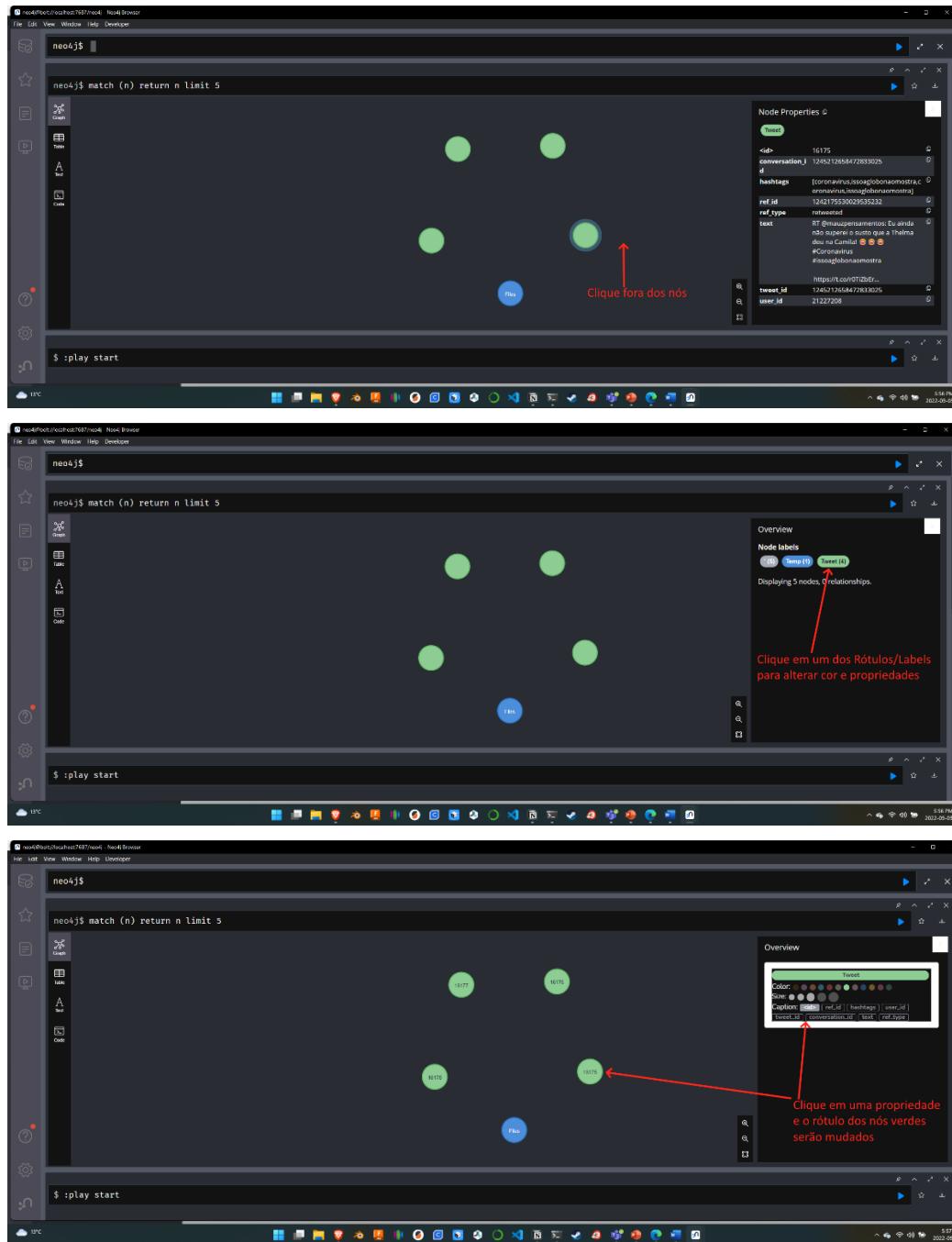
CALL apoc.load.directory('*.json')
YIELD value
WITH value as arquivos
ORDER BY arquivos DESC
CALL apoc.load.json(arquivos)
YIELD value
UNWIND value.data AS tweet
MERGE (t:Tweet {tweet_id: tweet.id})
SET t.text = tweet.text, t.user_id = tweet.author_id,
t.conversation_id=tweet.conversation_id
MERGE (u:User {user_id:tweet.author_id})
MERGE (t)<-[:TWEETOU]-(u)
FOREACH (hashtag IN tweet.entities.hashtags |
    MERGE (h:Hashtag {tag: apoc.text.replace(apoc.text.clean(hashtag.tag),
    '^a-zA-Z0-9]', '')})
    MERGE (h)-[r:ESTA_EM]->(t)
)
FOREACH (ref IN tweet.referenced_tweets |
    SET t.tipo_ref = coalesce(t.tipo_ref, []) + [ref.type], t.id_ref =
coalesce(t.id_ref, []) + [ref.id]
);
    
```

Observe que no comando acima não são criados os nós de hashtag e não é feito nenhum tratamento do dado das tags, como normalização (tudo em minúsculas). Use o comando apoc.text.replace() com o apoc.text.clean() para deixar todas as hashtags padronizadas e eliminar criação de nós semelhantes.

Exemplo: **#CasaNova** e **#casa\_nova** geram dois nós e significam a mesma ideia. Estes comandos ajudarão a padronizar esta situação.

## 6. Professor, como eu devo colocar meu RU nos prints dos grafos?

**R:** Crie um nó com Label RU e com propriedade ru=seu\_número. Isto criará um nó de cor diferente dos demais. Para alterar a legenda que aparece em determinado grupo de nós, faça como nas imagens abaixo.



Sendo assim, desejo a todos um ótimo aprendizado e nos vemos na tutoria.

Atenciosamente.

Professor Guilherme D Patriota.