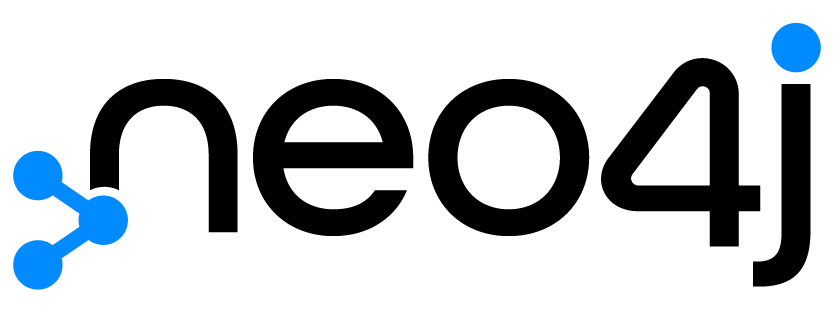
Tarea Neo4j



Guillermo de Carlos Muñoz

IABD 2024/2025

# 

[**Importación de datos 3**](#_kh0bqsfaf6c6)

[Preparación datos y entorno 3](#_gv0is1kcyweb)

[Subida de datos a Neo4j 6](#_e47gzsh5fukv)

[Business 6](#_wns0cv95nm0x)

[Checkin 7](#_a2yx9ejefyy2)

[Review 7](#_efb8nhf0pdti)

[Tip 8](#_a6w0hfvosnbm)

[User 9](#_rzv6p4uo0fqk)

[Relaciones 10](#_6qs89ficb2gq)

[User -> Review 10](#_gls4njermlqh)

[User -> Tip 10](#_c4a1haq7yu1v)

[Business -> Checkin 10](#_dcw2zbv01ile)

[Review -> Business 11](#_u8v6uy727jkk)

[**Aplicación de viajes 13**](#_o2zrlrwletsn)

[1. Encuentra los 10 hoteles con mayor número de reviews: 13](#_nynj5of6ulez)

[2. Encuentra los 10 usuarios con un número mayor de reviews realizadas. 13](#_b2h9lkwll24)

[**Consultoría empresa de viajes 14**](#_cx56gm75ugej)

[1. Encuentra los 50 usuarios con mayor número de reviews que han hecho una review del hotel Bellagio Hotel. 14](#_x3t8uwv6z62f)

[2. Buscar todos los hoteles que estos usuarios han hecho una review, no vale introducir los usuarios en el Where de manera manual. 15](#_n4relxu0vktj)

[3. Obtén el hotel con mayor número de reviews de los usuarios obtenidos en el apartado 1 de esta sección 15](#_gvf9dgvgysu3)

# 

# Importación de datos

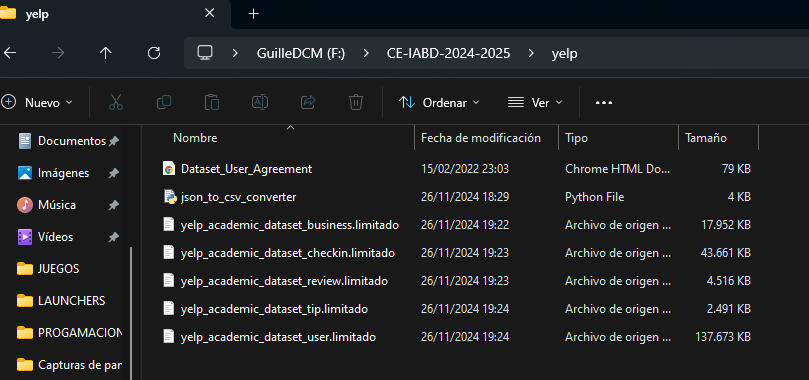
## Preparación datos y entorno

Lo primero que haremos será descargar el archivo de la página de yelp donde encontraremos los datos en formato json, un total de 5 archivos:

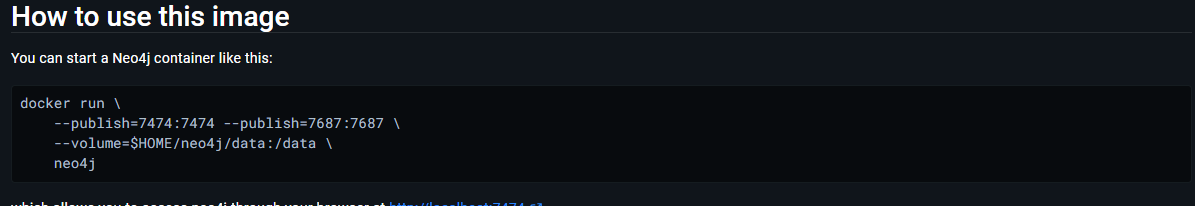
Al estar en formato json, usaremos un conversor de json a csv de python para trabajar mas fácilmente.

Una vez con los csv’s, vemos que son muy pesados y con muchos datos, como estamos aprendiendo, vamos a limitar estos csv´s, en mi caso a 10.000 instancias cada uno.

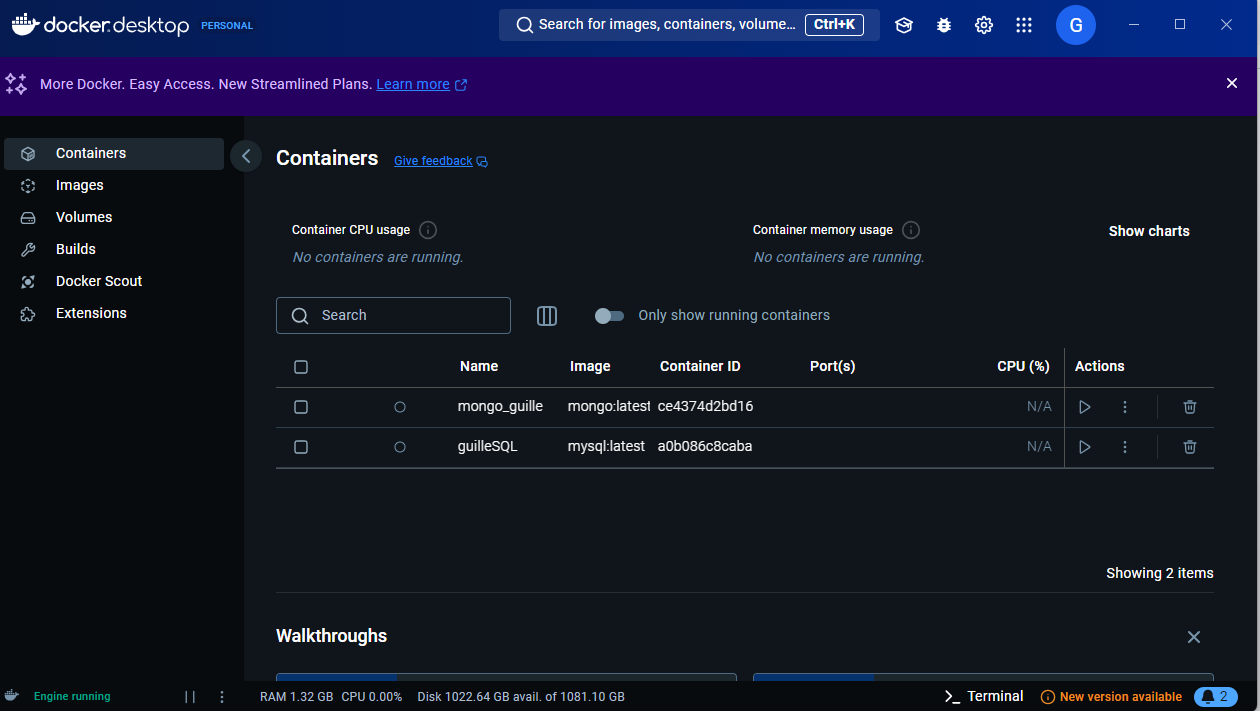
Tendremos entonces algo tal que asi:



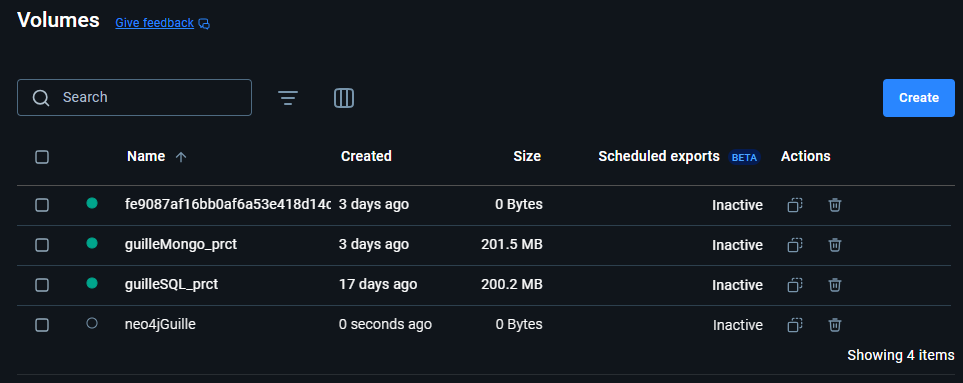
Por otro lado tenemos que preparar el container de docker con la imagen de neo4j, para ello, como siempre a docker hub <https://hub.docker.com/_/neo4j> .



Iniciamos docker, comprobamos las imágenes que tenemos:



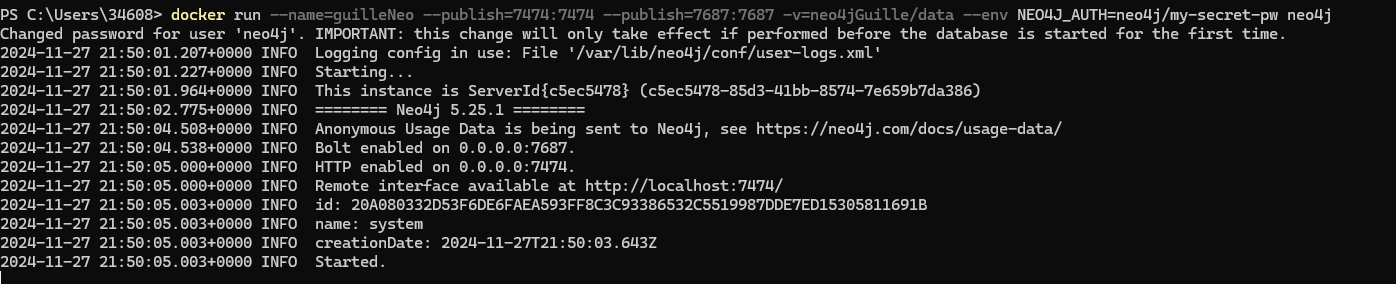
No está Neo4j

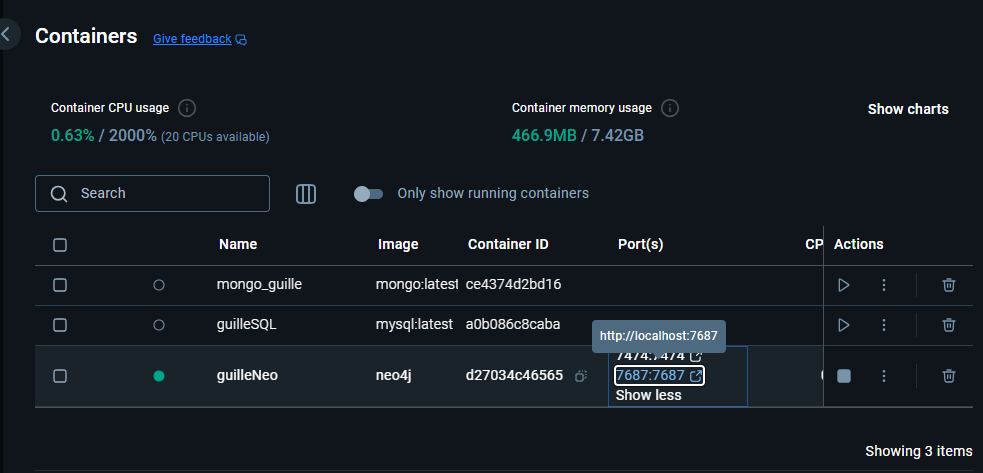


Creamos volumen antes del container, para asignarlo en la creación directamente.

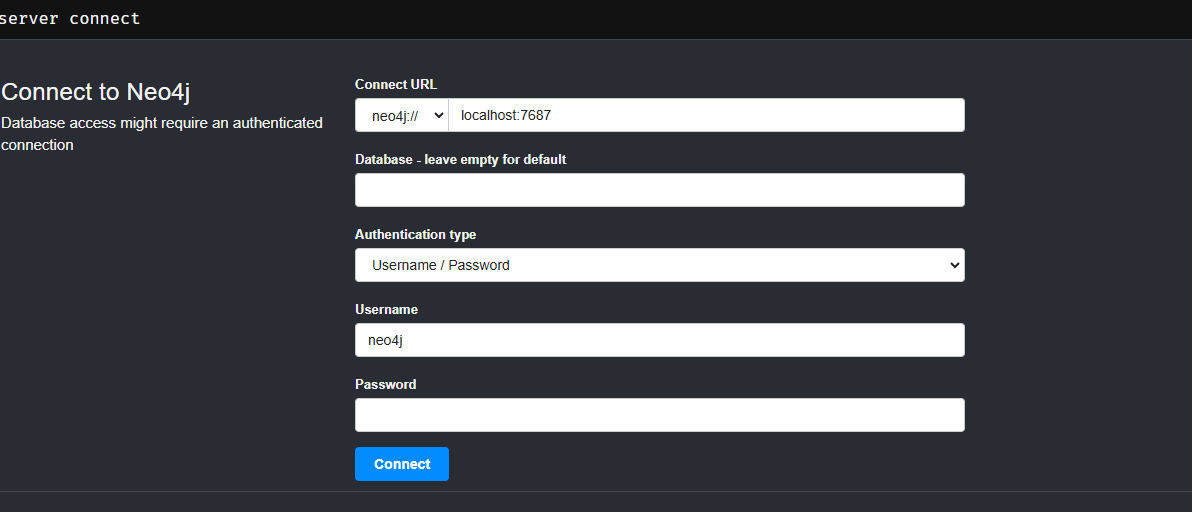
Creación de container en powershell:

docker run --name=guilleNeo --publish=7474:7474 --publish=7687:7687 -v=neo4jGuille/data --env NEO4J\_AUTH=neo4j/my-secret-pw neo4j





Ya esta creado!!, ahora nos metemos en el <http://localhost:7474/browser/> que es uno de los dos puertos que expone, en el que podremos ver todo de manera visual con una GUI.

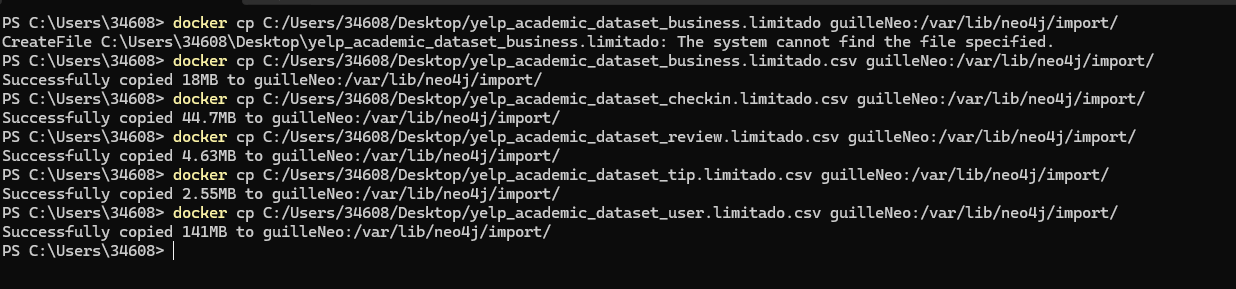


Nos logueamos con el usuario y contra que hemos puesto en el container.

*Mis compas me han chivado que en el csv de business es recomendable/obligatorio cambiar las comillas por espacios en blanco para evitar problemas a la hora de importarlo.*

Ya todo preparado para la subida a neo4j.

## Subida de datos a Neo4j

Primero tendremos que meter los csv´s al container con docker cp  


docker cp C:/Users/34608/Desktop/yelp\_academic\_dataset\_business.limitado.csv guilleNeo:/var/lib/neo4j/import/

docker cp C:/Users/34608/Desktop/yelp\_academic\_dataset\_checkin.limitado.csv guilleNeo:/var/lib/neo4j/import/

docker cp C:/Users/34608/Desktop/yelp\_academic\_dataset\_review.limitado.csv guilleNeo:/var/lib/neo4j/import/

docker cp C:/Users/34608/Desktop/yelp\_academic\_dataset\_tip.limitado.csv guilleNeo:/var/lib/neo4j/import/

docker cp C:/Users/34608/Desktop/yelp\_academic\_dataset\_user.limitado.csv guilleNeo:/var/lib/neo4j/import/

Ya metidos en el container, con el comando load csv , los cargamos en neo4j y montamos los nodos con sus atributos correspondientes:

### Business

LOAD CSV WITH HEADERS FROM "file:///yelp\_academic\_dataset\_business.limitado.csv" AS row

CREATE (b:Business)

SET b.business\_id = row.business\_id,

b.name = row.name,

b.address = row.address,

b.city = row.city,

b.state = row.state,

b.postal\_code = row.postal\_code,

b.latitude = toFloat(row.latitude),

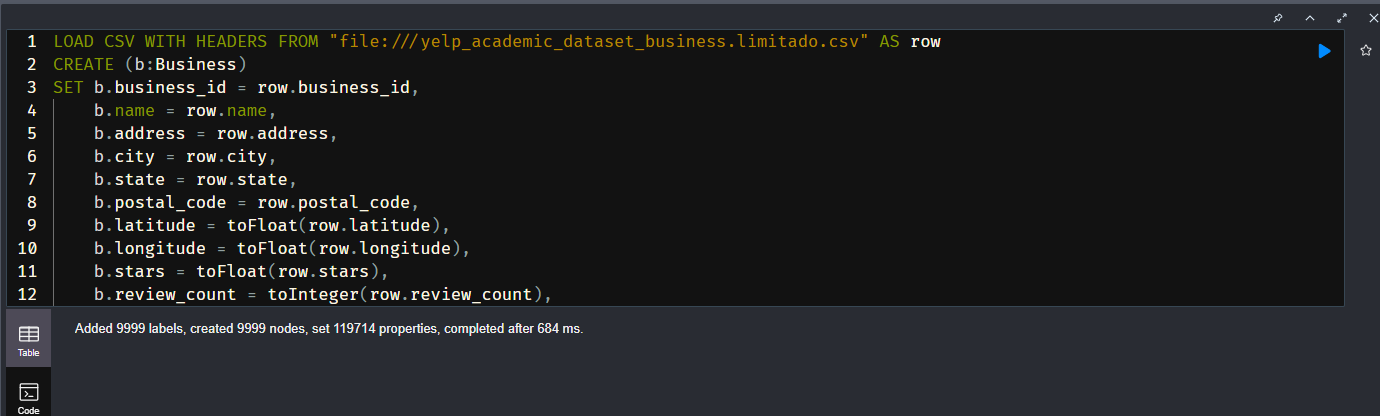
b.longitude = toFloat(row.longitude),

b.stars = toFloat(row.stars),

b.review\_count = toInteger(row.review\_count),

b.is\_open = (row.is\_open <> "0"),

b.categories = row.categories



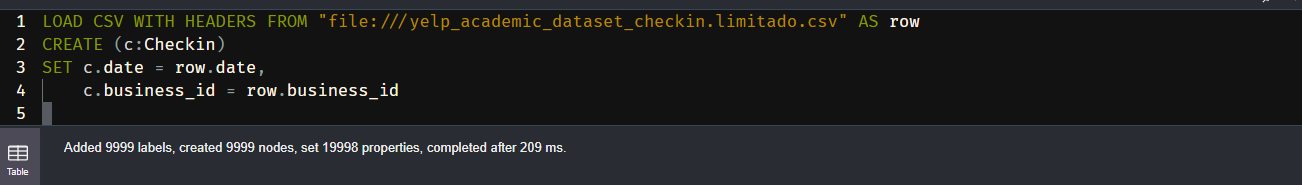
### Checkin

LOAD CSV WITH HEADERS FROM "file:///yelp\_academic\_dataset\_checkin.limitado.csv" AS row

CREATE (c:Checkin)

SET c.date = row.date,

c.business\_id = row.business\_id



### Review

LOAD CSV WITH HEADERS FROM "file:///yelp\_academic\_dataset\_review.limitado.csv" AS row

CREATE (r:Review)

SET r.date = row.date,

r.user\_id = row.user\_id,

r.funny = toInteger(row.funny),

r.stars = toFloat(row.stars),

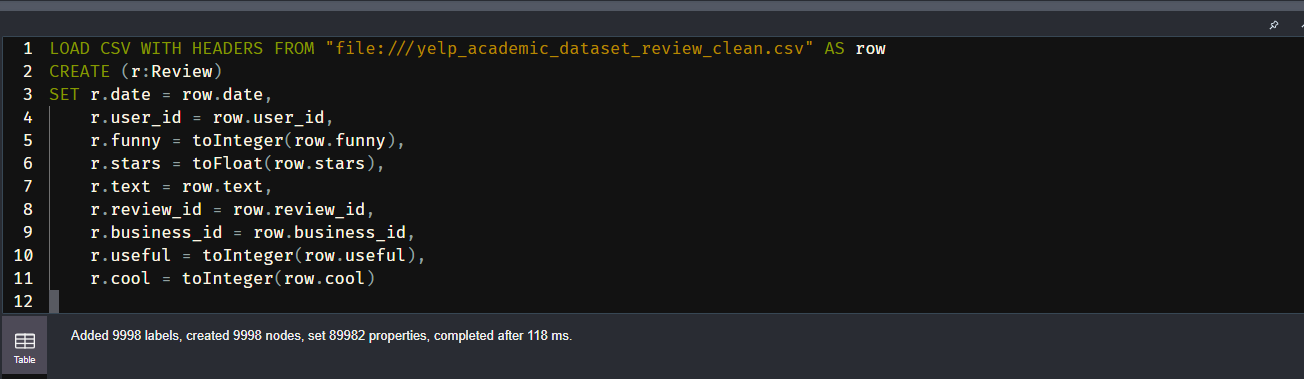
r.text = row.text,

r.review\_id = row.review\_id,

r.business\_id = row.business\_id,

r.useful = toInteger(row.useful),

r.cool = toInteger(row.cool)



En este de reviews he tenido problemas por tema de comillas abiertas sin en algunas estar cerradas, pero con un script de python lo he corregido y ha funcionado.

El script de py:

import csv

*# Abrir el archivo original y uno nuevo para escribir el archivo limpio*

with open('yelp\_academic\_dataset\_review.limitado.csv', 'r', newline='', encoding='latin1') as infile, \

open('yelp\_academic\_dataset\_review\_clean.csv', 'w', newline='', encoding='latin1') as outfile:

reader = csv.reader(infile, delimiter=',', quotechar='"')

writer = csv.writer(outfile, delimiter=',', quotechar='"', quoting=csv.QUOTE\_MINIMAL)

for row in reader:

*# Escribe las filas corregidas en el archivo limpio*

writer.writerow(row)

### Tip

LOAD CSV WITH HEADERS FROM "file:///yelp\_academic\_dataset\_tip.limitado.csv" AS row

CREATE (t:Tip)

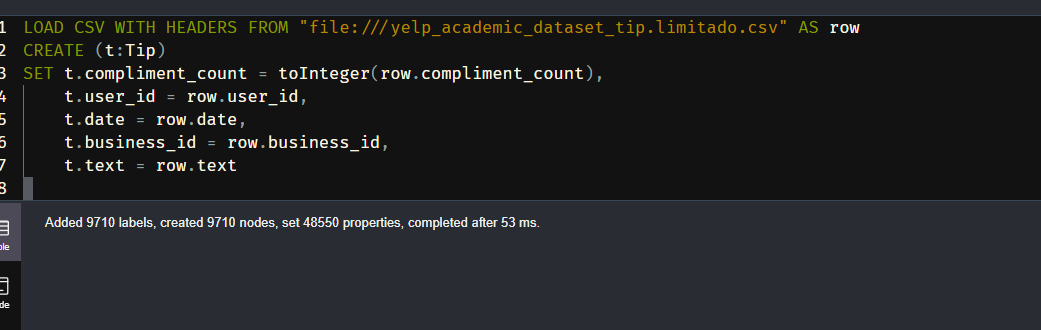
SET t.compliment\_count = toInteger(row.compliment\_count),

t.user\_id = row.user\_id,

t.date = row.date,

t.business\_id = row.business\_id,

t.text = row.text



### User

Los atributos del nodo de user me los voy a inventar ya que no me deja abrir el csv en drive, me dice que es muy grande xd

LOAD CSV WITH HEADERS FROM "file:///yelp\_academic\_dataset\_user.limitado.csv" AS row

CREATE (u:User)

SET u.user\_id = row.user\_id,

u.name = row.name,

u.review\_count = toInteger(row.review\_count),

u.yelping\_since = row.yelping\_since,

u.friends\_count = toInteger(row.friends\_count),

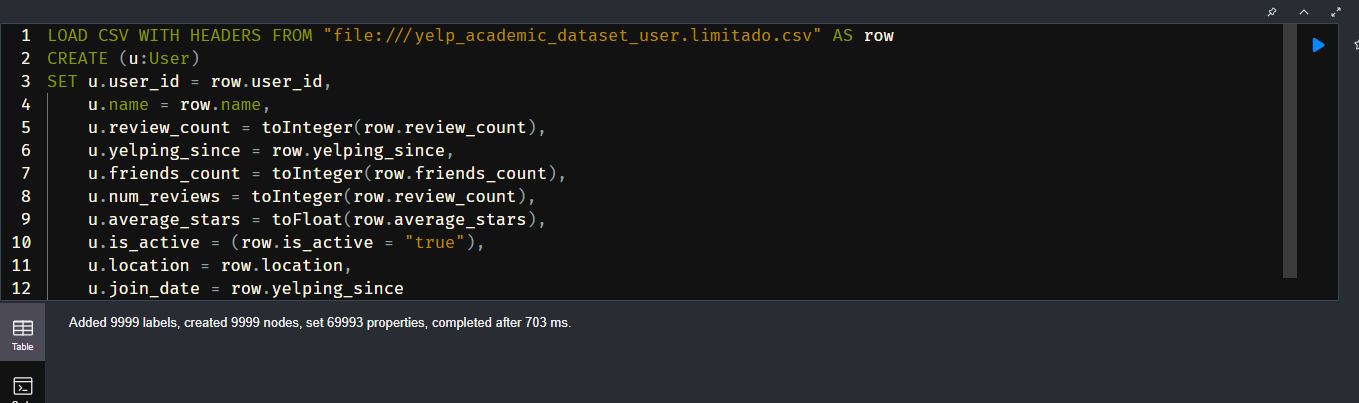
u.num\_reviews = toInteger(row.review\_count),

u.average\_stars = toFloat(row.average\_stars),

u.is\_active = (row.is\_active = "true"),

u.location = row.location,

u.join\_date = row.yelping\_since



Con estas 5 importaciones ya tendríamos todos los nodos creados.

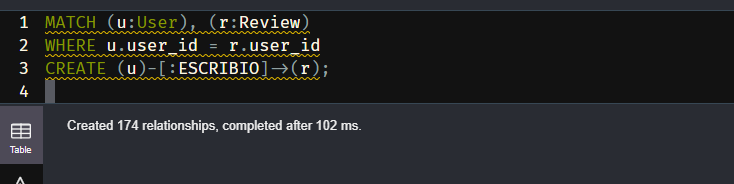
## Relaciones

### User -> Review

MATCH (u:User), (r:Review)

WHERE u.user\_id = r.user\_id

CREATE (u)-[:ESCRIBIO]->(r);

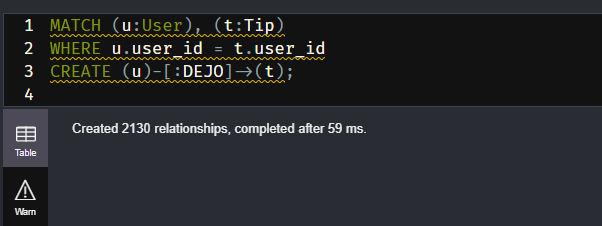


### User -> Tip

MATCH (u:User), (t:Tip)

WHERE u.user\_id = t.user\_id

CREATE (u)-[:DEJO]->(t);

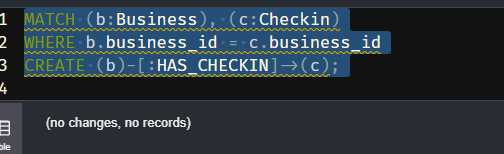


### Business -> Checkin

MATCH (b:Business), (c:Checkin)

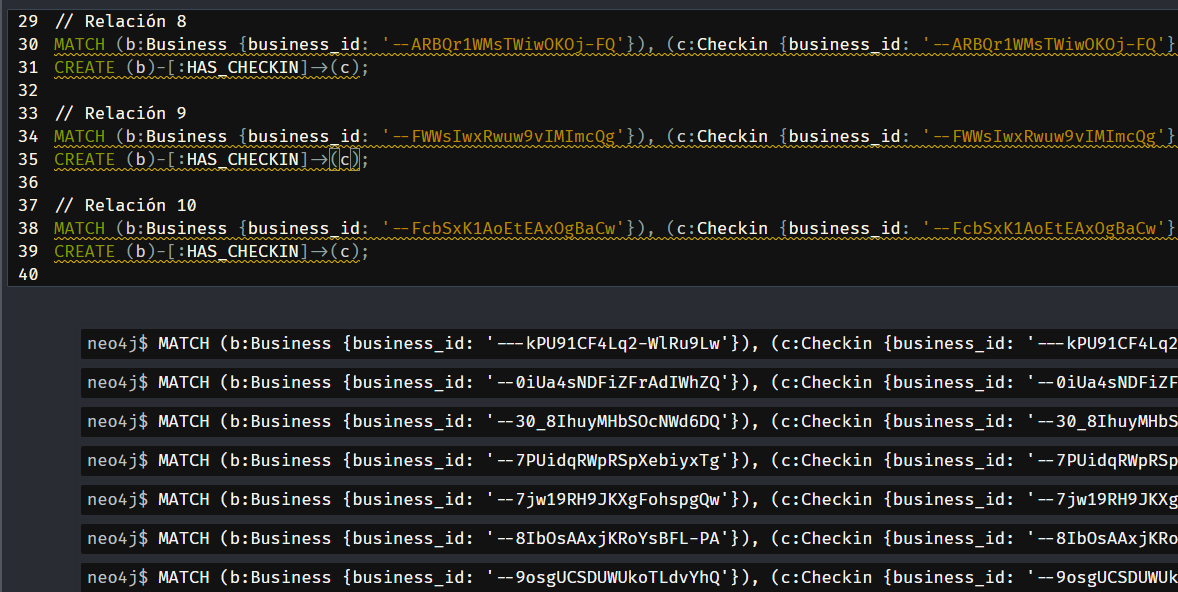
WHERE b.business\_id = c.business\_id

CREATE (b)-[:TIENE\_CHECKIN]->(c);



No hay relaciones para esta, las vamos a crear manualmente.

Crearlo manualmente las relaciones con, aqui por ejemplo 10



### Review -> Business

MATCH (r:Review), (b:Business)

WHERE r.business\_id = b.business\_id

CREATE (r)-[:CALIFICA]->(b);

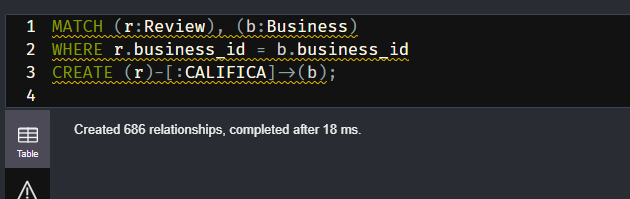
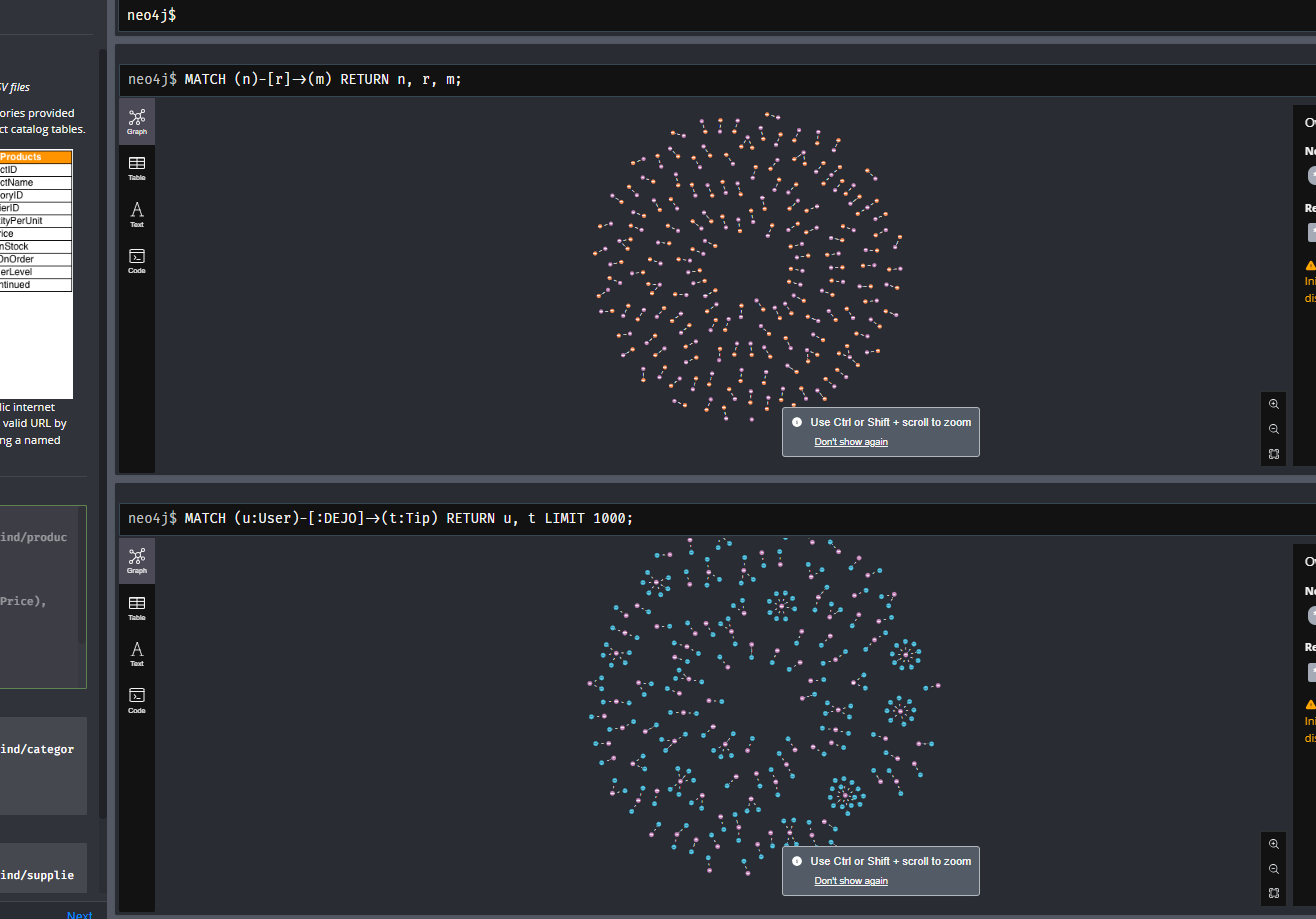


Imagen del grafo con diferentes relaciones



# Aplicación de viajes

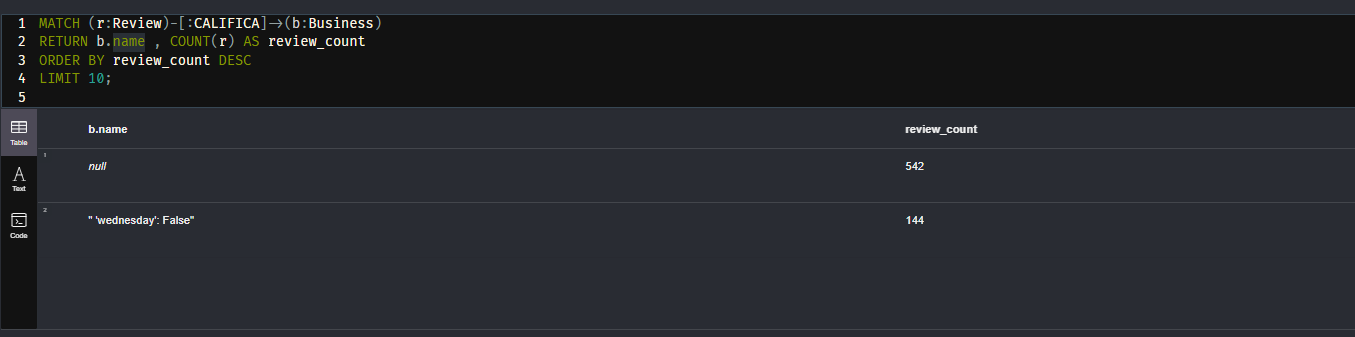
## 1. Encuentra los 10 hoteles con mayor número de reviews:

MATCH (r:Review)-[:CALIFICA]->(b:Business)

RETURN b.name , COUNT(r) AS review\_count

ORDER BY review\_count DESC

LIMIT 10;

  
Solo me devuelven 2 y uno null, esto sera porque al no meter todos los datos, puede que no coincidan los ids en las relaciones.

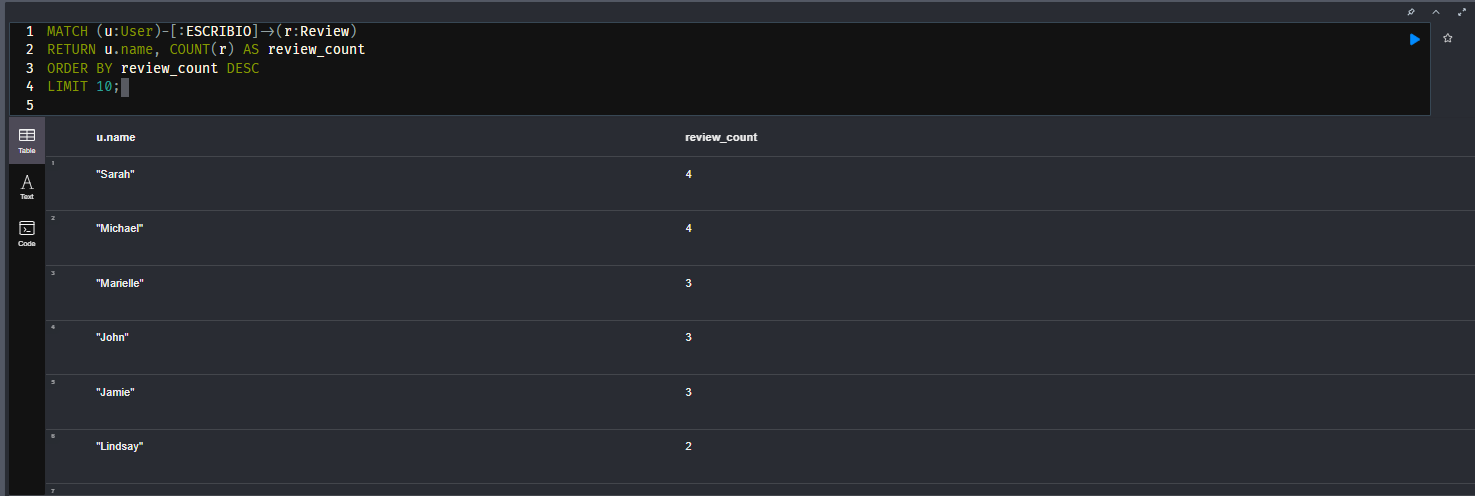
## 2. Encuentra los 10 usuarios con un número mayor de reviews realizadas.

MATCH (u:User)-[:ESCRIBIO]->(r:Review)

RETURN u.name, COUNT(r) AS review\_count

ORDER BY review\_count DESC

LIMIT 10;



Aqui vemos a los usuarios con el numero de reseñas ordenado descendente

# Consultoría empresa de viajes

## 1. Encuentra los 50 usuarios con mayor número de reviews que han hecho una review del hotel Bellagio Hotel.

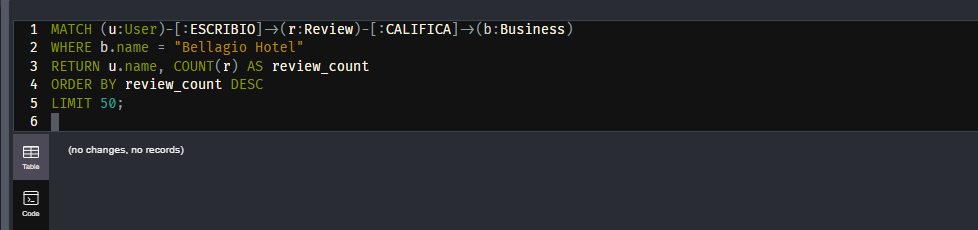
MATCH (u:User)-[:ESCRIBIO]->(r:Review)-[:CALIFICA]->(b:Business)

WHERE b.name = "Bellagio Hotel"

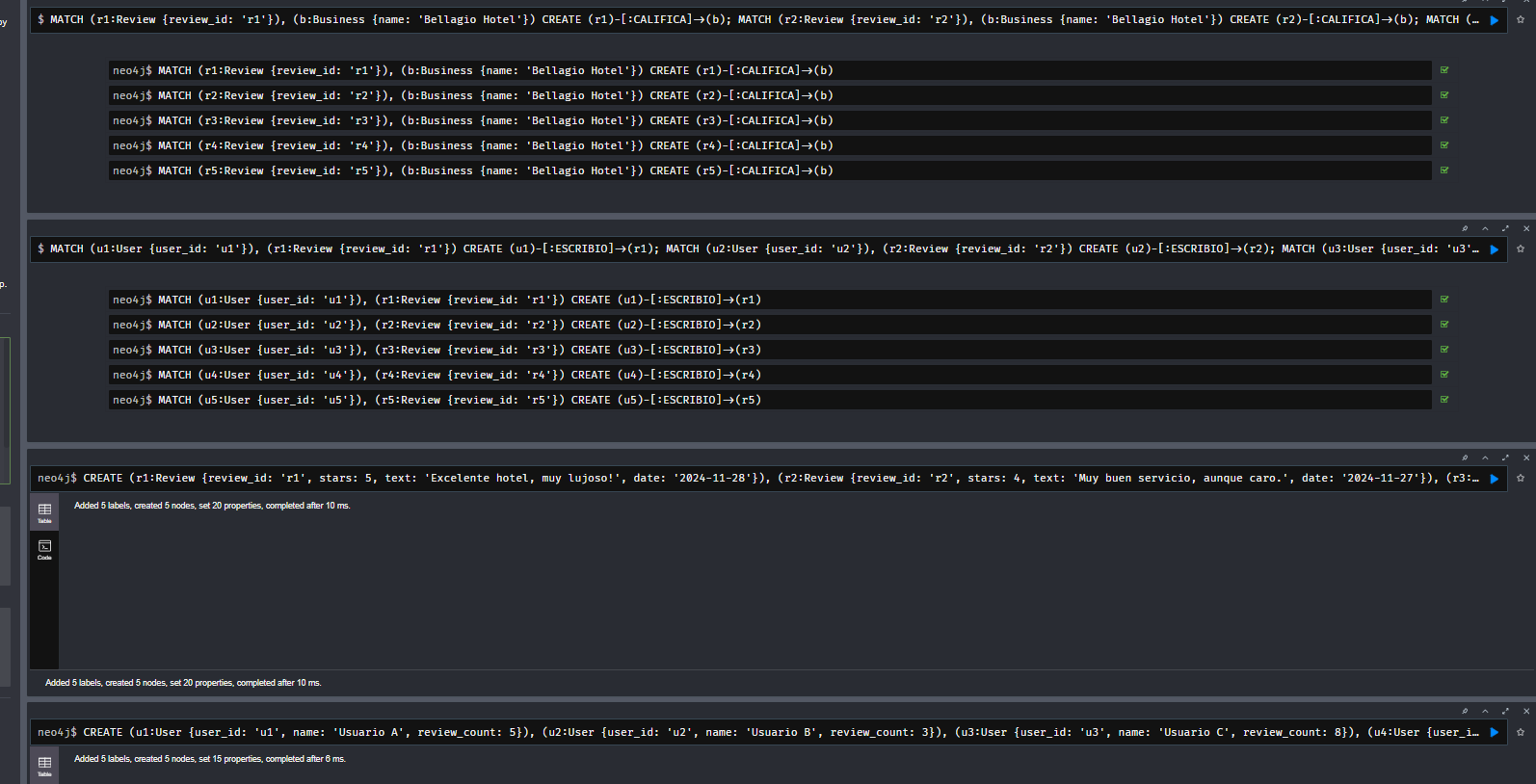
RETURN u.name, COUNT(r) AS review\_count

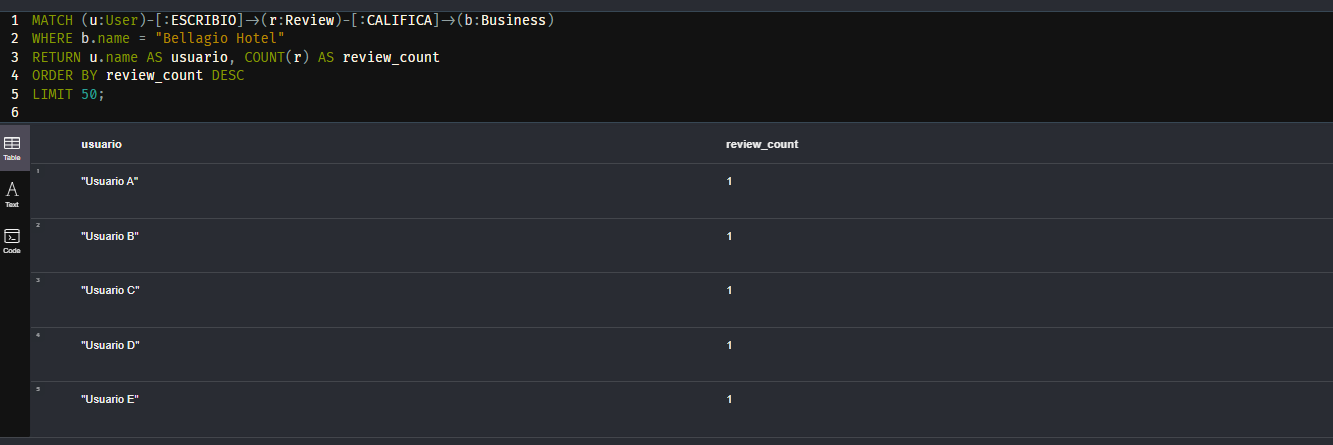
ORDER BY review\_count DESC

LIMIT 50;



Aqui de primeras no me salian datos , por lo que he metido datos de prueba para comprobar que esta bien la consulta:



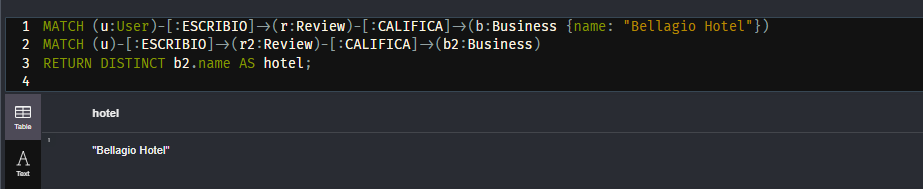


## 2. Buscar todos los hoteles que estos usuarios han hecho una review, no vale introducir los usuarios en el Where de manera manual.

MATCH (u:User)-[:ESCRIBIO]->(r:Review)-[:CALIFICA]->(b:Business {name: "Bellagio Hotel"})

MATCH (u)-[:ESCRIBIO]->(r2:Review)-[:CALIFICA]->(b2:Business)

RETURN DISTINCT b2.name AS hotel;



Primero encuentra todos los usuariosque han escrito una reseña sobre el "Bellagio Hotel" (filtrando el buisiners por name: "Bellagio Hotel").

Y luego que los mismos usuarios (u) que reseñaron el "Bellagio Hotel" también han reseñado en general.

## 3. Obtén el hotel con mayor número de reviews de los usuarios obtenidos en el apartado 1 de esta sección

MATCH (u:User)-[:ESCRIBIO]->(r:Review)-[:CALIFICA]->(b:Business {name: "Bellagio Hotel"})

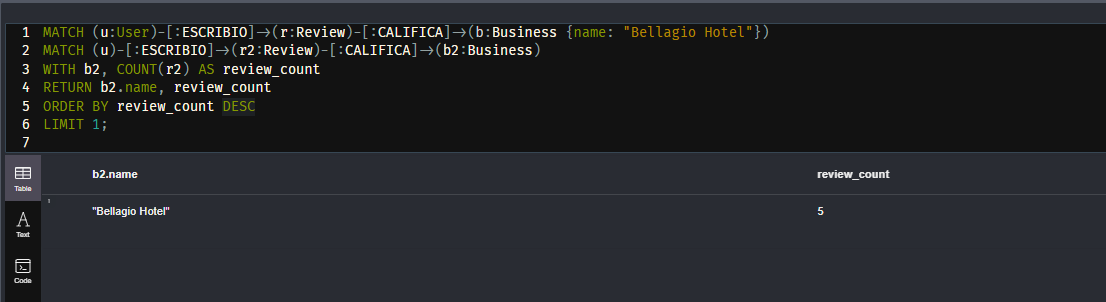
MATCH (u)-[:ESCRIBIO]->(r2:Review)-[:CALIFICA]->(b2:Business)

WITH b2, COUNT(r2) AS review\_count

RETURN b2.name, review\_count

ORDER BY review\_count DESC

LIMIT 1;



La dos últimas si que las he visto más complicadas, pero aun asi lo veo mas facil que mongo xd.