

Tarea Neo4J

Importación de datos

Para empezar, una vez descargado los datos los exporto empleando el comando tar.

```
PS C:\Users\Vespertino> tar -xvf Downloads/yelp_dataset.tar
x Dataset_User_Agreement.pdf
x yelp_academic_dataset_business.json
x yelp_academic_dataset_checkin.json
x yelp_academic_dataset_review.json
x yelp_academic_dataset_tip.json
x yelp_academic_dataset_user.json
PS C:\Users\Vespertino>
```

Una vez exportados, con un programa de Python cambio su formato de json a csv.

```
C:\Users\Vespertino>py json_to_csv_converter.py yelp_academic_dataset_business.json
C:\Users\Vespertino>py json_to_csv_converter.py yelp_academic_dataset_checkin.json
C:\Users\Vespertino>py json_to_csv_converter.py yelp_academic_dataset_review.json
C:\Users\Vespertino>py json_to_csv_converter.py yelp_academic_dataset_user.json
C:\Users\Vespertino>py json_to_csv_converter.py yelp_academic_dataset_tip.json
-
```

El primer intento del ejercicio fue empleando solo los primeros 2000 campos de cada csv, pero debido a la falta de relaciones decidí volver a intentarlo usando todos los datos.

Por culpa del enorme tamaño de “user” y “review”, me he visto obligado a recortar estos dos csv a sus primeros 100000 para este intento.

```
PS C:\Users\WINDOWS 11\Documents\csv> Get-Content ".\yelp_academic_dataset_review.csv" | select -First 100000 | Out-File ".\yelp_academic_dataset_review_recortado2.csv"
PS C:\Users\WINDOWS 11\Documents\csv> Get-Content ".\yelp_academic_dataset_user.csv" | select -First 100000 | Out-File ".\yelp_academic_dataset_review_user2.csv"
```

Mando los archivos recortados al interior del contenedor empleando docker cp.

```
PS C:\Users\Vespertino\csv> docker cp ".\out" d80473fd6a79:/var/lib/neo4j/import/.
Successfully copied 58.2MB to d80473fd6a79:/var/lib/neo4j/import/.
PS C:\Users\Vespertino\csv>
```

Lo siguiente ha sido importar los datos de la siguiente forma dentro de neo4j, y tras ello he creado las siguientes relaciones.

```
LOAD CSV WITH HEADERS FROM 'file:///yelp_academic_dataset_business_out.csv' AS row
CREATE (n:Business)
SET n = row

LOAD CSV WITH HEADERS FROM 'file:///yelp_academic_dataset_checkin_out.csv' AS row
CREATE (n:Checkin)
SET n = row

LOAD CSV WITH HEADERS FROM 'file:///yelp_academic_dataset_review_out.csv' AS row
CREATE (n:Review)
SET n = row

LOAD CSV WITH HEADERS FROM 'file:///yelp_academic_dataset_user_out.csv' AS row
CREATE (n:User)
SET n = row

LOAD CSV WITH HEADERS FROM 'file:///yelp_academic_dataset_tip_out.csv' AS row
CREATE (n:Tip)
SET n = row
```

Los Usuarios escriben Reviews que están destinadas a Negocios

```
MATCH (r:Review),(u:User)
WHERE r.user_id = u.user_id
CREATE (u)-[:OPINA]->(r)

MATCH (b:Business),(r:Review)
WHERE b.business_id = r.business_id
CREATE (r)-[:Califica]->(b)
```

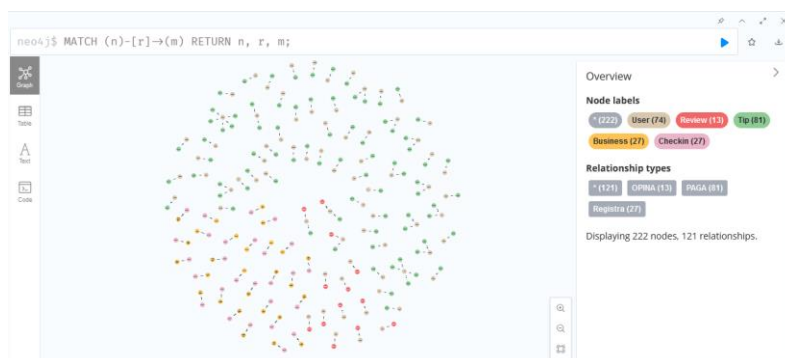
Los Negocios realizan Checkins

```
MATCH (b:Business),(c:Checkin)
WHERE b.business_id = c.business_id
CREATE (b)-[:Registra]->(c)
```

Y los Usuarios escriben Consejos

```
MATCH (t:Tip),(u:User)
WHERE t.user_id = u.user_id
CREATE (u)-[:PAGA]->(t)
```

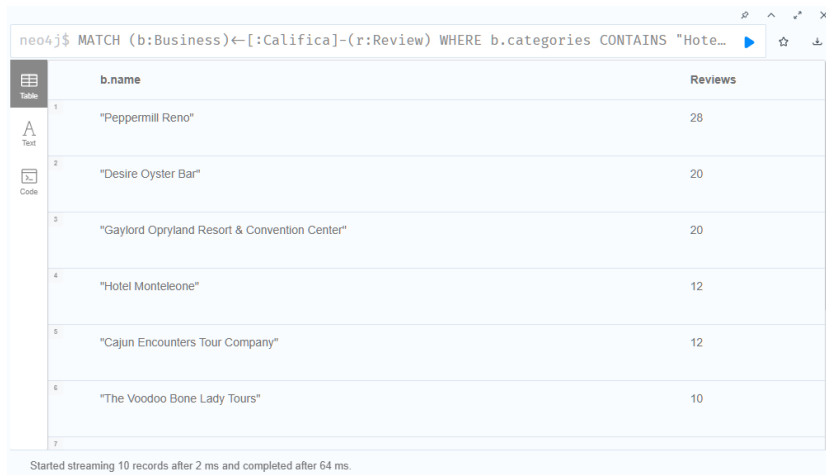
(la relación es “PAGA” porque pensaba que se refería a Tip de “propina”)



Aplicación de Viaje

Encuentra los 10 hoteles con mayor número de reviews.

```
MATCH (b:Business)<-[:Califica]-(r:Review)
WHERE b.categories CONTAINS "Hotel"
RETURN b.name, count(r) AS Reviews
ORDER BY Reviews DESC
LIMIT 10;
```



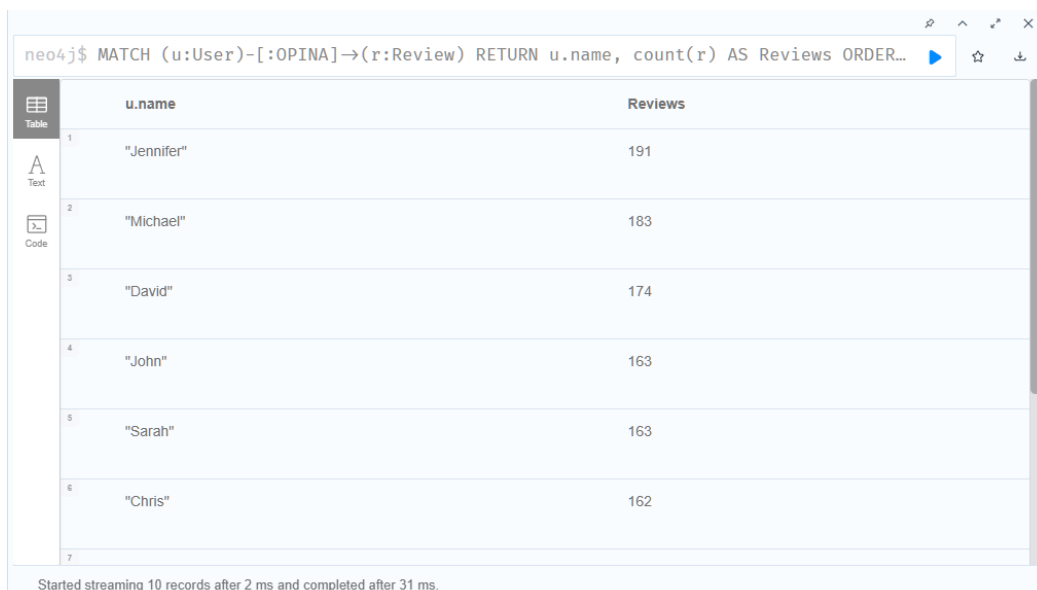
The screenshot shows the Neo4j query interface with the following Cypher query: `MATCH (b:Business)<-[:Califica]-(r:Review) WHERE b.categories CONTAINS "Hotel" RETURN b.name, count(r) AS Reviews ORDER BY Reviews DESC LIMIT 10;` The results are displayed in a table with two columns: `b.name` and `Reviews`. The table lists 10 hotels, with the first being "Peppermill Reno" with 28 reviews. The interface includes a sidebar with icons for Table, Text, and Code, and a status bar at the bottom indicating the query execution time.

	b.name	Reviews
1	"Peppermill Reno"	28
2	"Desire Oyster Bar"	20
3	"Gaylord Opryland Resort & Convention Center"	20
4	"Hotel Monteleone"	12
5	"Cajun Encounters Tour Company"	12
6	"The Voodoo Bone Lady Tours"	10
7		

Started streaming 10 records after 2 ms and completed after 64 ms.

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

```
MATCH (u:User)-[:OPINA]->(r:Review)
RETURN u.name, count(r) AS Reviews
ORDER BY Reviews DESC
LIMIT 10;
```



The screenshot shows the Neo4j query interface with the following Cypher query: `MATCH (u:User)-[:OPINA]->(r:Review) RETURN u.name, count(r) AS Reviews ORDER BY Reviews DESC LIMIT 10;` The results are displayed in a table with two columns: `u.name` and `Reviews`. The table lists 10 users, with the first being "Jennifer" with 191 reviews. The interface includes a sidebar with icons for Table, Text, and Code, and a status bar at the bottom indicating the query execution time.

	u.name	Reviews
1	"Jennifer"	191
2	"Michael"	183
3	"David"	174
4	"John"	163
5	"Sarah"	163
6	"Chris"	162
7		

Started streaming 10 records after 2 ms and completed after 31 ms.

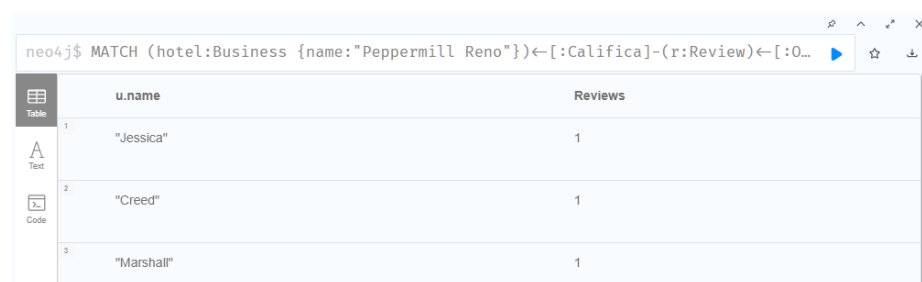
Consultoría empresa de viajes

A partir de aquí es posible que salgan errores debido a la falta de datos de User y Review. Estas son las siguientes consultas.

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

He cambiado el Bellagio por el Reno para mejor funcionamiento.

```
1 MATCH (hotel:Business {name:"Peppermill Reno"})←[:Califica]-(r:Review)←
  [:OPINA]-(u:User)
2 WITH u, count(r) AS Reviews
3 RETURN u.name, Reviews
4 ORDER BY Reviews DESC
5 LIMIT 50;
```



	u.name	Reviews
1	"Jessica"	1
2	"Creed"	1
3	"Marshall"	1

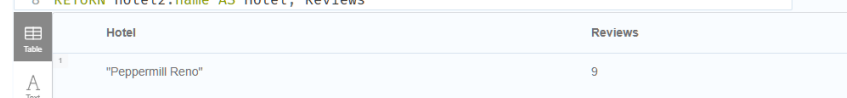
2. Buscar todos los hoteles que estos usuarios han hecho una review, no vale introducir los usuarios en el Where de manera manual.

```
1 MATCH (hotel:Business {name:"Peppermill Reno"})←[:Califica]-(r:Review)←[:OPINA]-(u:User)
2 WITH u
3 MATCH (u)-[:OPINA]→(r2:Review)-[:Califica]→(hotel2:Business)
4 WHERE hotel2.categories CONTAINS "Hotel" and hotel2.name <> "Peppermill Reno"
5 WITH hotel2, count(r2) AS Reviews
6 ORDER BY Reviews DESC
7 LIMIT 50
8 RETURN hotel2.name AS Hotel, Reviews
```

(no changes, no records)

Si se permite el mismo hotel la consulta funciona.

```
1 MATCH (hotel:Business {name:"Peppermill Reno"})←[:Califica]-(r:Review)←
  [:OPINA]-(u:User)
2 WITH u
3 MATCH (u)-[:OPINA]→(r2:Review)-[:Califica]→(hotel2:Business)
4 WHERE hotel2.categories CONTAINS "Hotel"
5 WITH hotel2, count(r2) AS Reviews
6 ORDER BY Reviews DESC
7 LIMIT 50
8 RETURN hotel2.name AS Hotel, Reviews
```



	Hotel	Reviews
1	"Peppermill Reno"	9

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

```
1 MATCH (hotel:Business {name:"Peppermill Reno"})←[:Califica]-(r:Review)←[:OPINA]-(u:User)
2 WITH u
3 MATCH (u)-[:OPINA]→(r2:Review)-[:Califica]→(hotel2:Business)
4 WHERE hotel2.categories CONTAINS "Hotel" AND hotel2.name <> "Peppermill Reno"
5 WITH hotel2, count(r2) AS Reviews
6 ORDER BY Reviews DESC
7 LIMIT 1
8 RETURN hotel2.name AS Hotel, Reviews;
```

(no changes, no records)

Como en el apartado anterior, si se elimina la exclusión, funciona.

```
1 MATCH (hotel:Business {name:"Peppermill Reno"})←[:Califica]-(r:Review)←[:OPINA]-(u:User)
2 WITH u
3 MATCH (u)-[:OPINA]→(r2:Review)-[:Califica]→(hotel2:Business)
4 WHERE hotel2.categories CONTAINS "Hotel"
5 WITH hotel2, count(r2) AS Reviews
6 ORDER BY Reviews DESC
7 LIMIT 1
8 RETURN hotel2.name AS Hotel, Reviews;
```

	Hotel	Reviews
1	"Peppermill Reno"	9