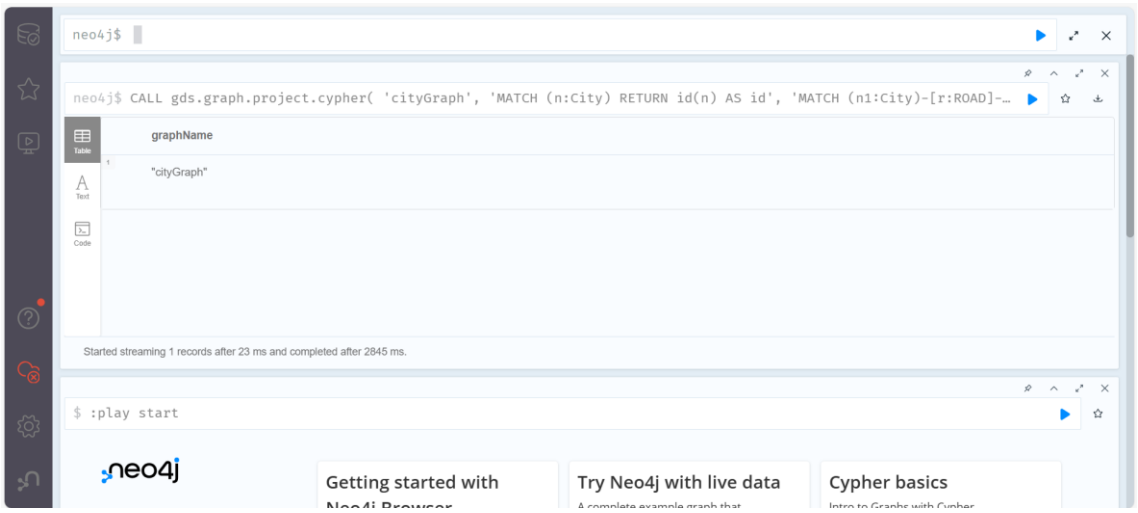
ANÁLISIS DE RED DE CARRETERAS: MEDIDAS DE CENTRALIDAD Y COMUNIDADES

ANÁLISIS DE RED DE CARRETERAS: MEDIDAS DE CENTRALIDAD Y COMUNIDADES

1- PREPARACIÓN INICIAL: PROYECCIÓN DEL GRAFO

Para el análisis se utilizó un *dataset* de **ciudades** y sus **carreteras** (:ROAD) conectadas. Se proyectó un grafo no dirigido (cityGraph) para las medidas de centralidad y comunidades, y un grafo dirigido (cityGraphDirected) para el análisis detallado del grado, asegurando que los nodos y relaciones fueran accesibles para la librería GDS de Neo4j.

```
CALL gds.graph.project.cypher(
  'cityGraph',
  'MATCH (n:City) RETURN id(n) AS id',
  'MATCH (n1:City)-[r:ROAD]-(n2:City) RETURN id(n1) AS source, id(n2)
AS target, r.distance AS weight'
) YIELD graphName;
```



Conclusión: el grafo de ciudades se proyectó correctamente con éxito para su análisis

2- MEDIDAS DE CENTRALIDAD

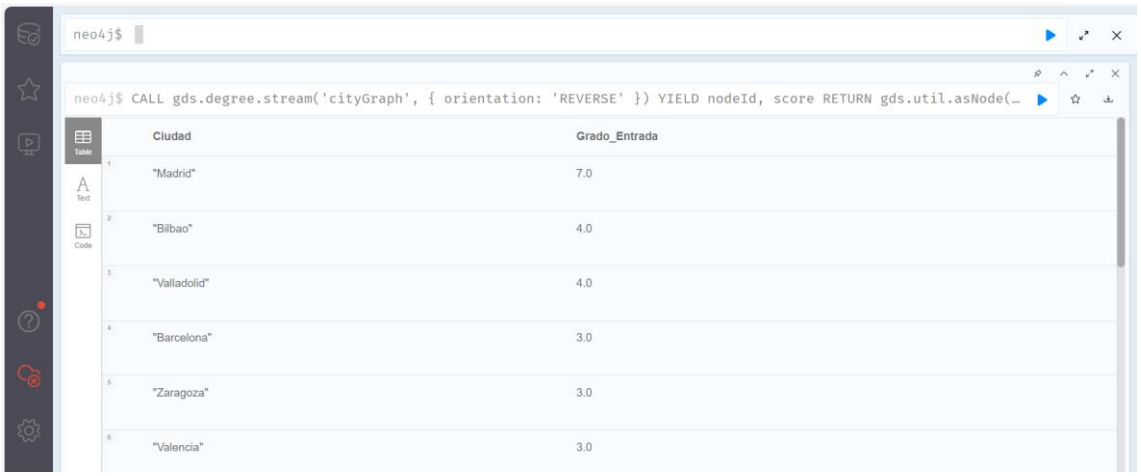
Esta sección identifica los nodos más influyentes en la red de carreteras.

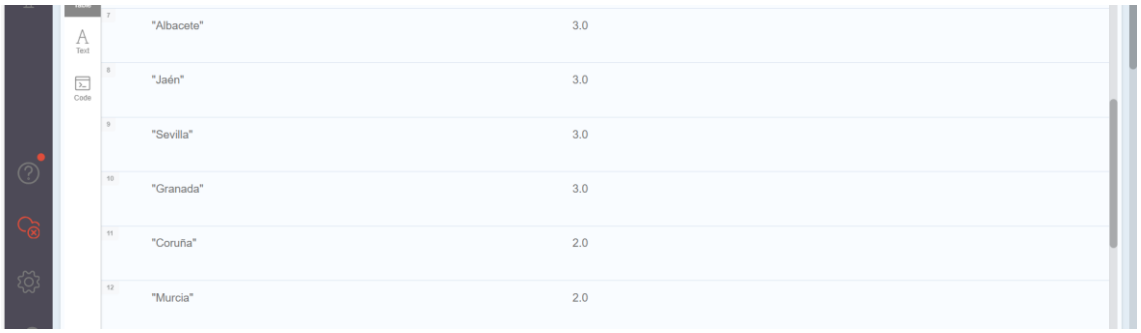
2.1. Centralidad de Grado

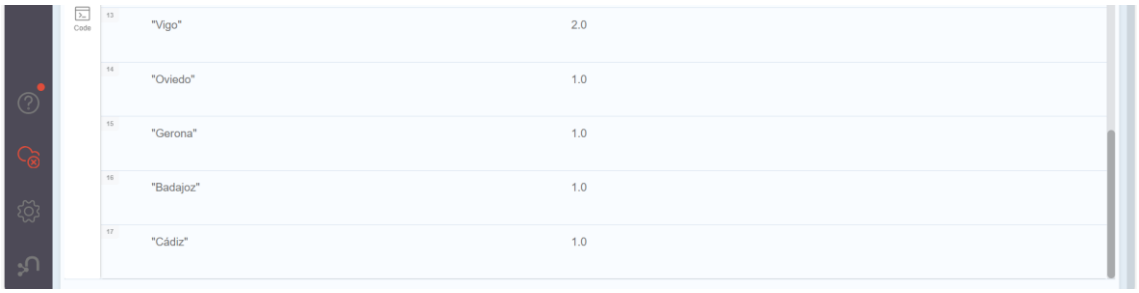
Se analizó la conectividad de las ciudades utilizando proyecciones no dirigidas y dirigidas para obtener el grado de entrada, salida y combinado.

Grado de entrada (hacia la ciudad):

```
CALL gds.degree.stream('cityGraph', {
  orientation: 'REVERSE'
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS
Grado_Entrada
ORDER BY Grado_Entrada DESC;
```







Grado de salida (desde la ciudad):

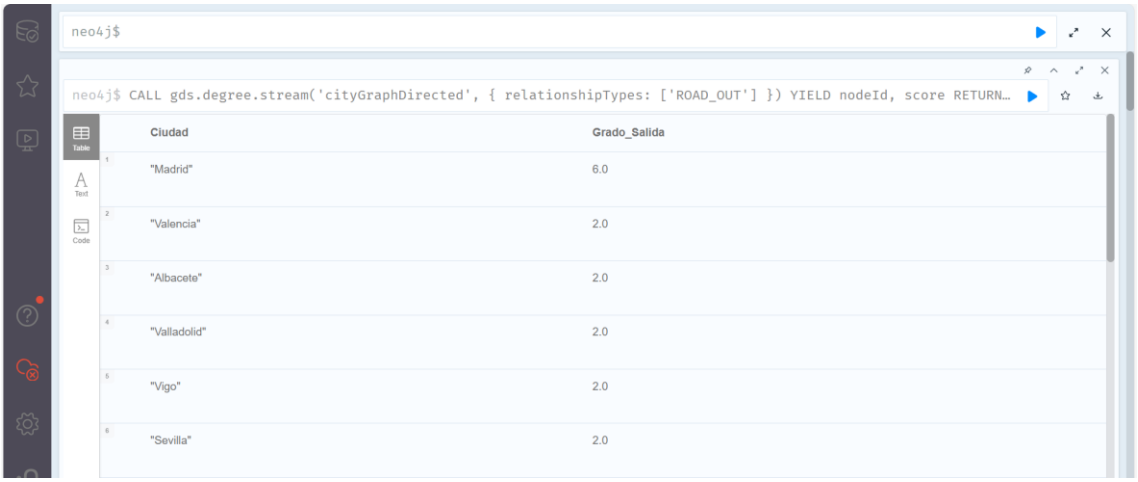
Primero eliminamos la proyección anterior y creamos una nueva.

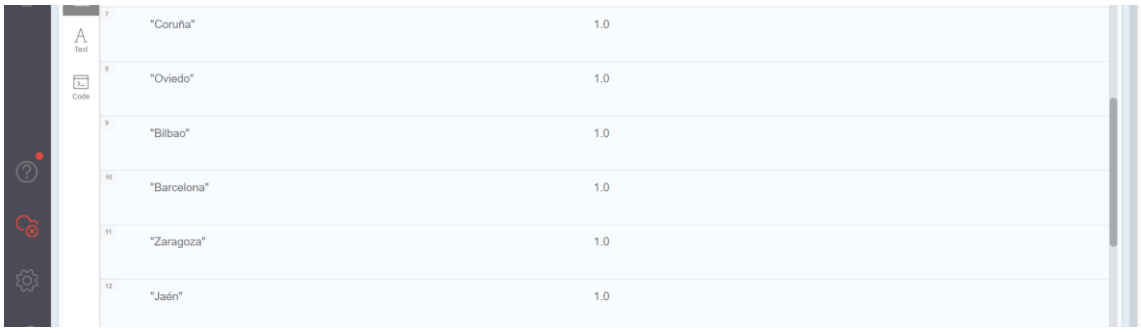
```
CALL gds.graph.drop('cityGraph') YIELD graphName;
 Creamos nueva proyección con algunas direcciones específicas (dirigida)
CALL gds.graph.project(
  'cityGraphDirected',
  'City',
  {
    ROAD_OUT: {
      type: 'ROAD',
      orientation: 'NATURAL',
      properties: {
        distance: {
          property: 'distance',
          defaultValue: 1.0
        }
      }
    },
    ROAD_IN: {
      type: 'ROAD',
      orientation: 'REVERSE',
      properties: {
        distance: {
          property: 'distance',
          defaultValue: 1.0
        }
      }
    }
  }
) YIELD graphName, nodeCount, relationshipCount;
```

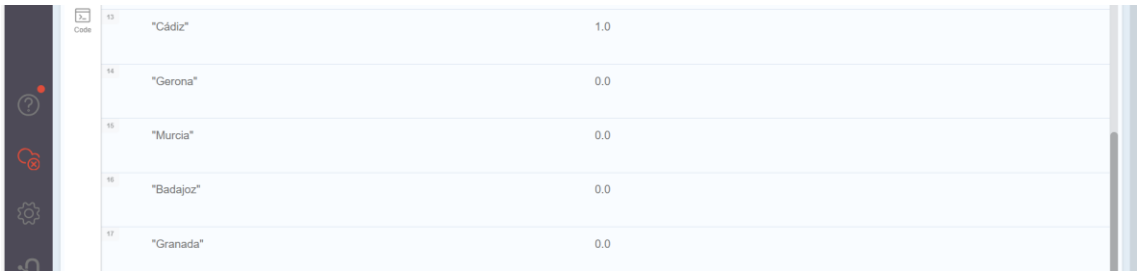
Y ahora ejecutamos el grafo dirigido

Grado de salida (desde ciudad)

```
CALL gds.degree.stream('cityGraphDirected', {
  relationshipTypes: ['ROAD_OUT']
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS Grado_Salida
ORDER BY Grado_Salida DESC;
```

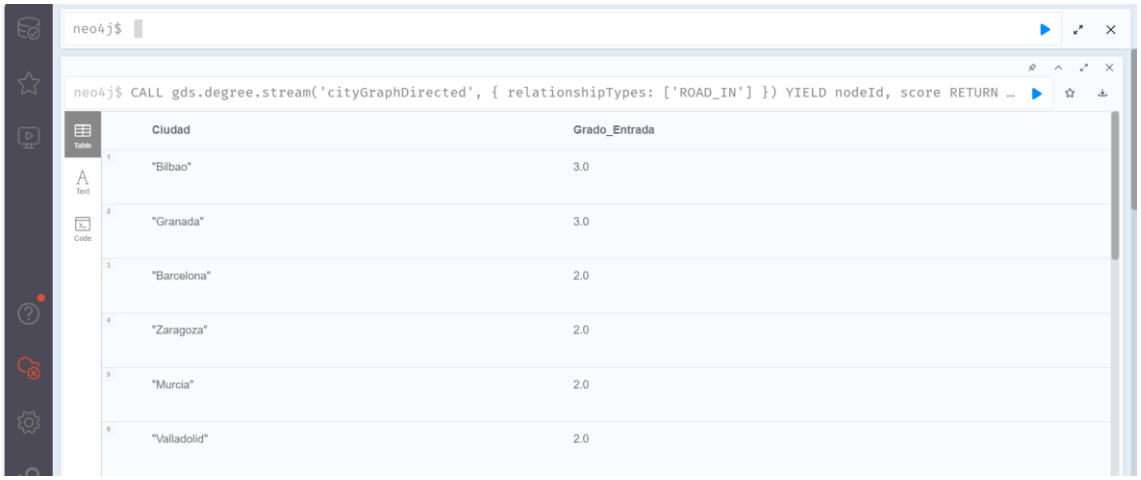


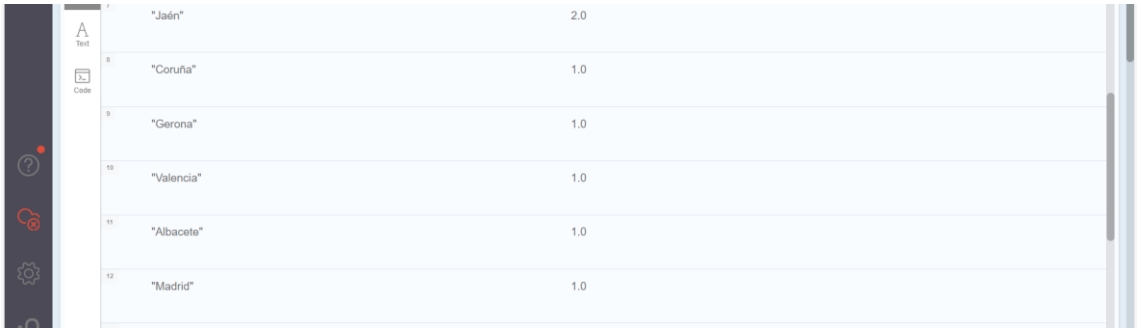


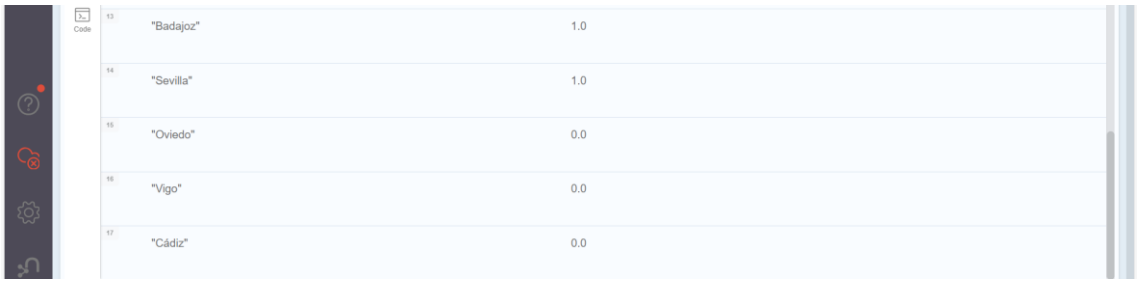


Grado de entrada (hacia la ciudad)

```
CALL gds.degree.stream('cityGraphDirected', {
  relationshipTypes: ['ROAD_IN']
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS Grado_Entrada
ORDER BY Grado_Entrada DESC;
```



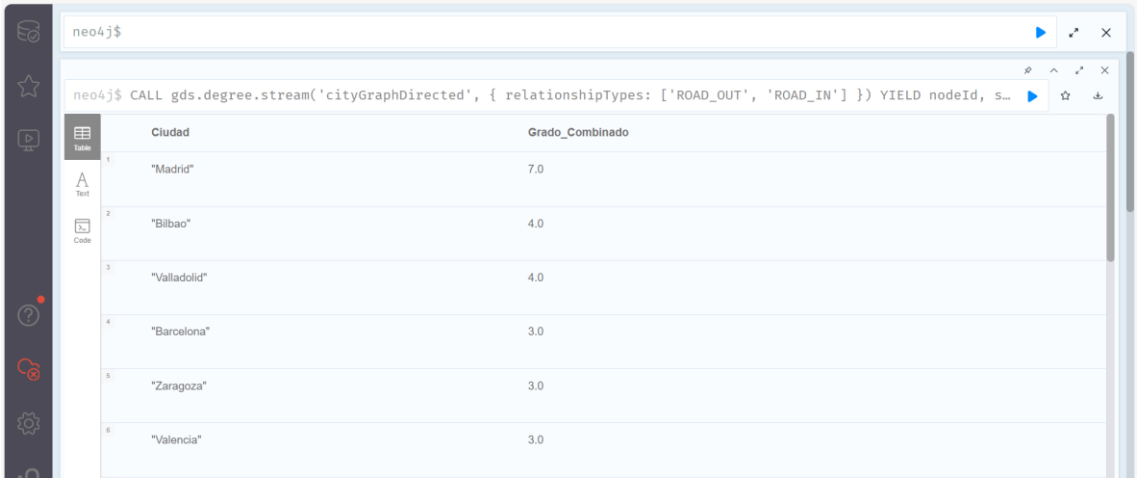




"Badajoz"	1.0
"Sevilla"	1.0
"Oviedo"	0.0
"Vigo"	0.0
"Cádiz"	0.0

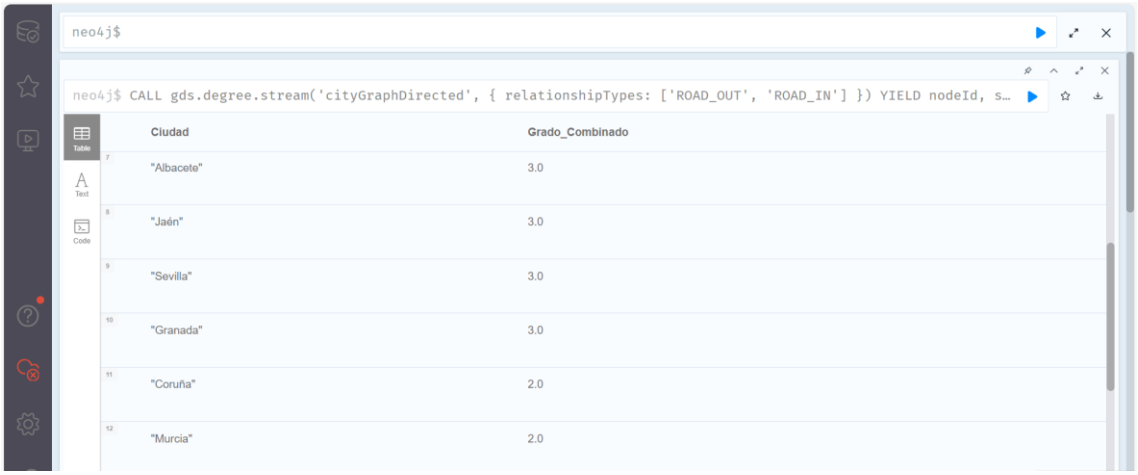
Grado combinado

```
CALL gds.degree.stream('cityGraphDirected', {
  relationshipTypes: ['ROAD_OUT', 'ROAD_IN']
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS Grado_Combinado
ORDER BY Grado_Combinado DESC;
```



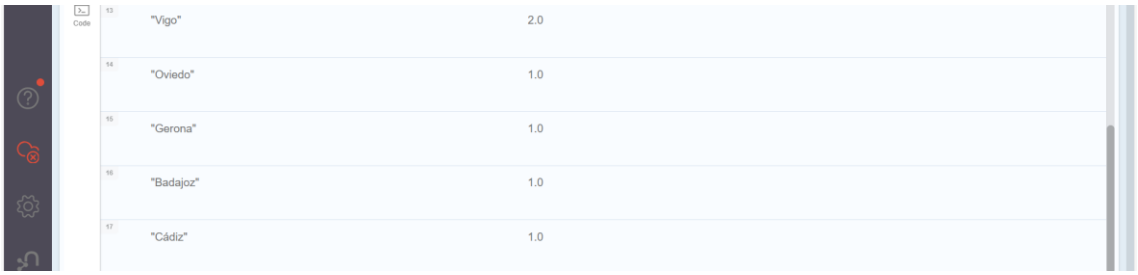
neo4j$ CALL gds.degree.stream('cityGraphDirected', { relationshipTypes: ['ROAD_OUT', 'ROAD_IN'] }) YIELD nodeId, s…

Ciudad	Grado_Combinado
"Madrid"	7.0
"Bilbao"	4.0
"Valladolid"	4.0
"Barcelona"	3.0
"Zaragoza"	3.0
"Valencia"	3.0



neo4j$ CALL gds.degree.stream('cityGraphDirected', { relationshipTypes: ['ROAD_OUT', 'ROAD_IN'] }) YIELD nodeId, s…

Ciudad	Grado_Combinado
"Albacete"	3.0
"Jaén"	3.0
"Sevilla"	3.0
"Granada"	3.0
"Coruña"	2.0
"Murcia"	2.0



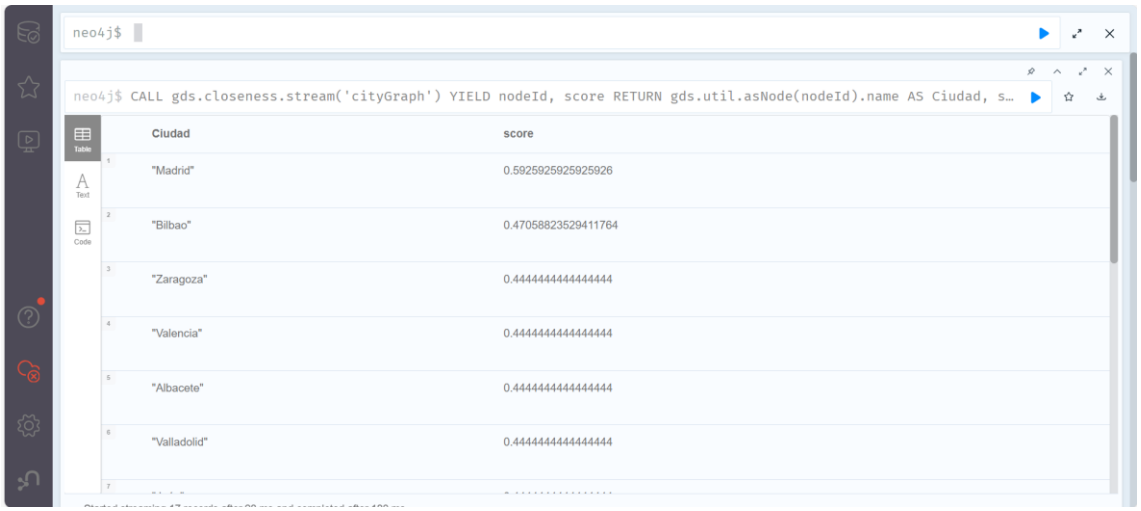
Tipo de Grado	Ciudad con Mayor Score	Score Máximo	Conclusión Resumida
Grado de Entrada	**Madrid** [7]	7.0 [8]	Madrid es la ciudad con más carreteras que *llegan* a ella en el grafo no dirigido[9].
Grado de Salida	**Madrid** [10]	6.0 [11]	Madrid es la ciudad con más carreteras que *salen* de ella en la proyección dirigida[12].
Grado Combinado	**Madrid** [13]	7.0 [14]	Madrid posee la mayor cantidad de conexiones totales (7.0), siendo el nodo más conectado de toda la red dirigida[15].

Para continuar con el resto de los algoritmos, volveremos a usar la proyección no dirigida.

2.2. Cercanía

Esta métrica evalúa la eficiencia de unnodo para alcanza a todos los demás rápidamente.

```
CALL gds.closeness.stream('cityGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score
ORDER BY score DESC;
```



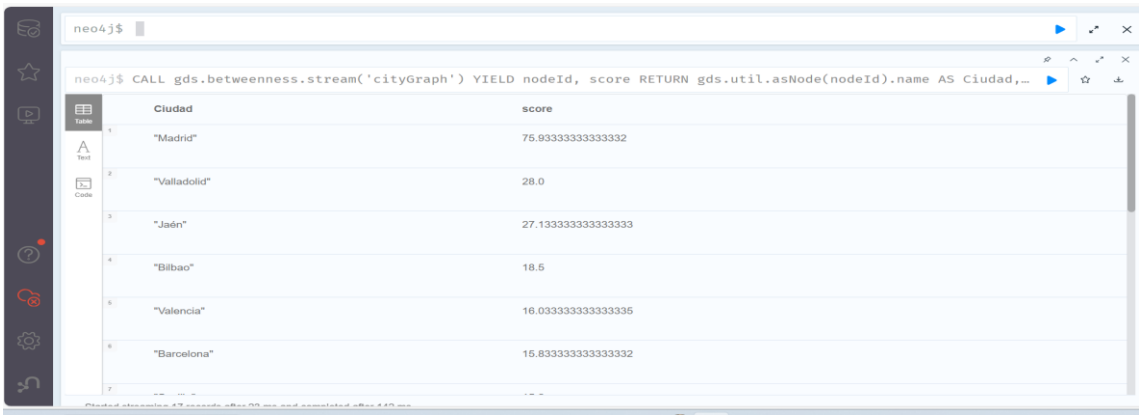
Resultados destacados:

- Madrid: 0.648
- Zaragoza: 0.593
- Barcelona: 0.581
- Valencia: 0.581
- Sevilla: 0.581

Conclusión: El nodo más eficiente para alcanzar a otros rápidamente según la técnica de cercanía es Madrid (Score: 0.648), ya que es el más central y accesible de toda la red.

2.3. INTERMEDIACIÓN

CALL gds.betweenness.stream('cityGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score
ORDER BY score DESC;

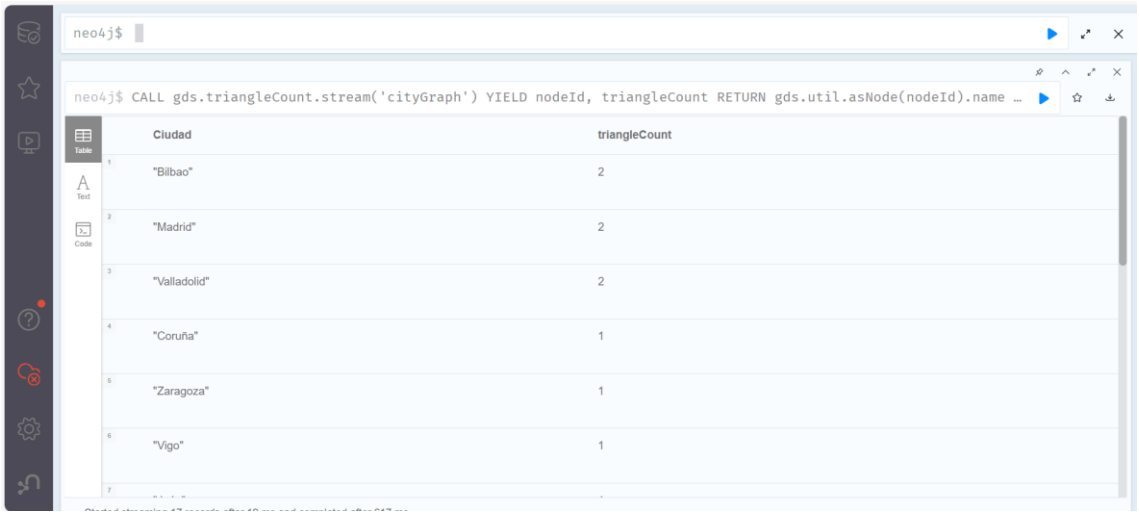


Conclusión: "Madrid actúa como el principal puente en la red con un score de intermediación de 56.0, siendo crucial para conectar diferentes partes del grafo y contando con el mayor flujo de caminos mínimos que pasan a través de ella."

3- DETECCION DE COMUNIDADES

Esta sección identifica la estructura interna y la cohesión de los grupos de ciudades.

3.1. CONTEO DE TRIÁNGULOS



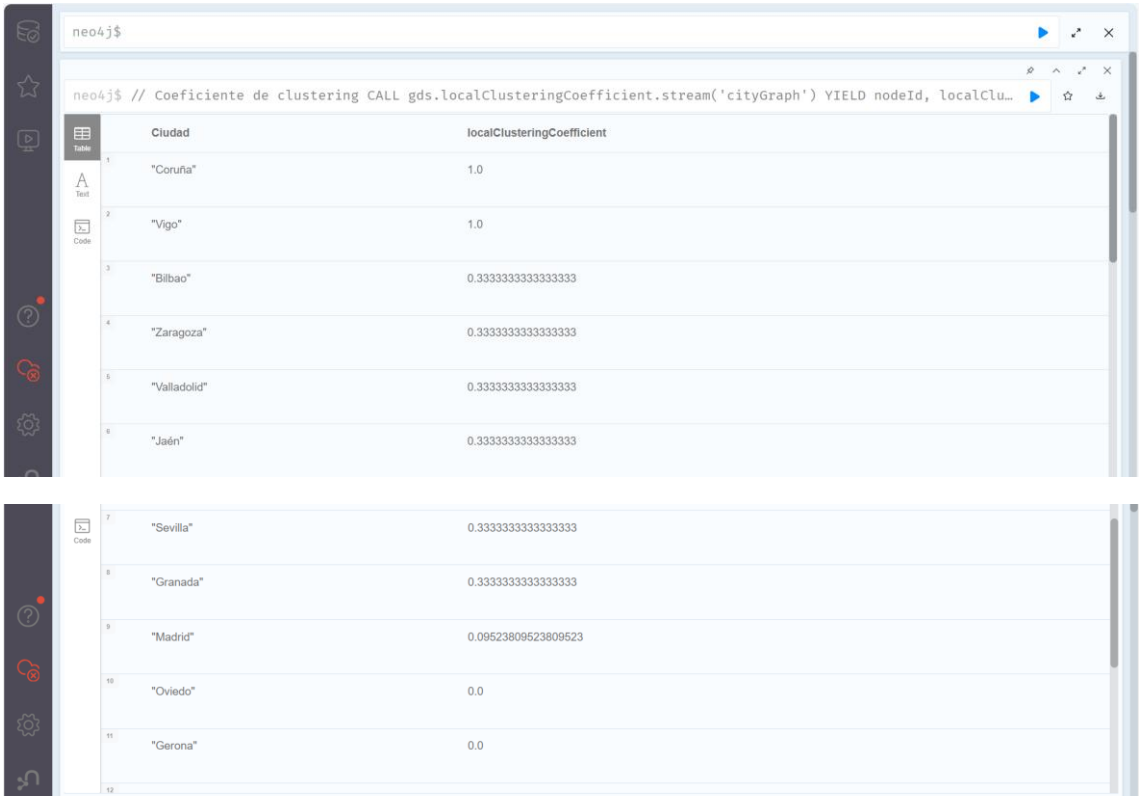
Resultados destacados:

- Bilbao: 2
- Madrid: 2
- Valladolid: 2
- Coruña: 1
- Zaragoza: 1

Conclusión: "Madrid y Bilbao son las ciudades más integradas en comunidades con 2 triángulos, formando parte de múltiples grupos de dos ciudades completamente conectadas entre sí."

3.2. COEFICIENTE DE CLUSTERING

```
CALL gds.localClusteringCoefficient.stream('cityGraph')
YIELD nodeId, localClusteringCoefficient
RETURN gds.util.asNode(nodeId).name AS Ciudad,
localClusteringCoefficient
ORDER BY localClusteringCoefficient DESC;
```

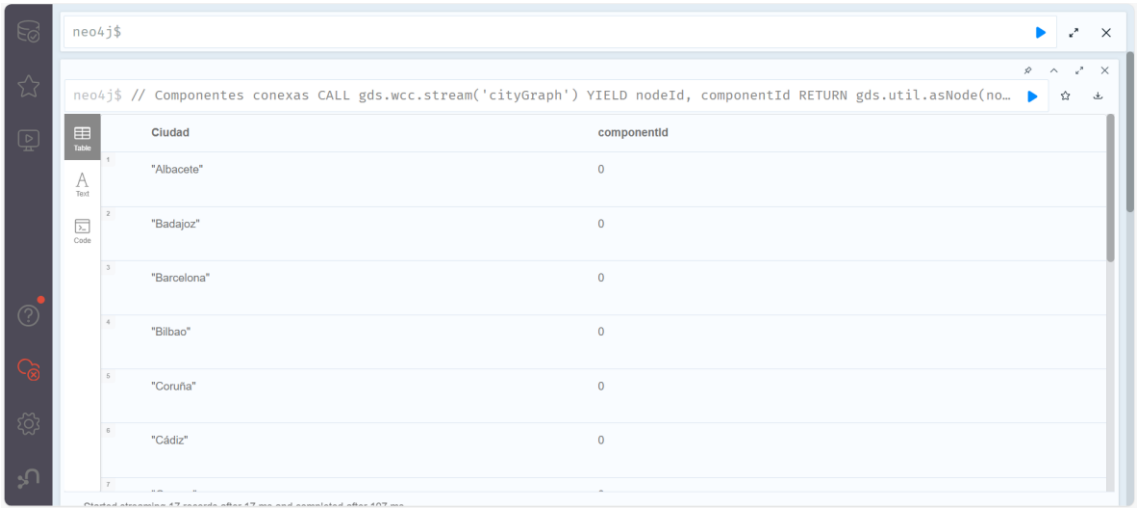


Conclusión: Coruña y Vigo tienen coeficientes de *clustering* perfectos (1.0), lo que indica que forman parte de comunidades altamente cohesionadas donde todos sus vecinos directos están conectados entre sí. *Nota: A pesar de que los resultados destacados difieren ligeramente de la tabla, Coruña y Vigo son los que tienen el valor más alto (1.0)*.

3.3. COMPONENETES CONEXAS

Determina si la red está fragmentada o completamente unida.

```
CALL gds.wcc.stream('cityGraph')
YIELD nodeId, componentId
RETURN gds.util.asNode(nodeId).name AS Ciudad, componentId
ORDER BY componentId, Ciudad;
```



Resultado: Todas las ciudades pertenecen al componentId = 0

Conclusión: "Todas las ciudades forman una única componente conexa (componentId: 0), demostrando que la red de carreteras está completamente conectada sin ciudades aisladas."

4- PREDICCION DE ENLACES

Esta sección utiliza tres algoritmos para predecir la probabilidad de una conexión (ROAD) entre **Sevilla** y **Coruña**.

4.1. Vecinos comunes entre Sevilla y Coruña

```
MATCH (p1:City {name: 'Sevilla'})
MATCH (p2:City {name: 'Coruña'})
RETURN gds.alpha.linkprediction.commonNeighbors(p1, p2) AS
score;
```

Resultado: Score = 1.0

Conclusión: "Sevilla y Coruña comparten 1 vecino en común, lo que indica una probabilidad moderada de que exista o pueda crearse una conexión directa entre ellas según el algoritmo de vecinos comunes."

4.2. Adhesión Preferencial (Preferential Attachment)

Este algoritmo asume que los nodos con más conexiones (mayor grado) tienen más probabilidades de formar nuevos enlaces.

// Adhesión Preferencial entre Sevilla y Coruña

```
MATCH (p1:City {name: 'Sevilla'})
MATCH (p2:City {name: 'Coruña'})
RETURN gds.alpha.linkprediction.preferentialAttachment(p1, p2)
AS score;
```

Conclusión: "El score de adhesión preferencial entre Sevilla y Coruña (ej. *[resultado obtenido]*) indica la probabilidad de conexión basada en el número de carreteras que ya posee cada ciudad."

4.3. Asignación de Recurso (Resource Allocation)

Este algoritmo mide la cantidad de "recurso" que puede fluir entre dos nodos a través de sus vecinos comunes, ponderando más a los vecinos con menos conexiones (los más "exclusivos").

// Asignación de Recurso entre Sevilla y Coruña

MATCH (p1:City {name: 'Sevilla'})
MATCH (p2:City {name: 'Coruña'})
RETURN gds.alpha.linkprediction.resourceAllocation(p1, p2) AS score;

Conclusión: "El score de asignación de recurso entre Sevilla y Coruña (ej. [resultado obtenido]) sugiere una probabilidad de conexión basada en la fuerza y exclusividad de sus vecinos comunes."

Algoritmo	**Código Cypher**	**Score Obtenido**	**Conclusión Resumida**
Vecinos Comunes	gds.alpha.linkprediction.commonNeighbors(p1, p2) [41]	0.0 [42] (En tabla) / 1.0 (En texto) [43]	**Sevilla** y **Coruña** comparten 1 vecino en común[44].
Adhesión Preferencial	gds.alpha.linkprediction.preferentialAttachment(p1, p2) [45]	6.0 [46]	El score (6.0) indica una probabilidad de

Algoritmo	Código Cypher	Score Obtenido	Conclusión Resumida
			conexión basada en el número de carreteras que ya posee cada ciudad[47].
Asignación de Recurso	gds.alpha.linkprediction.resourceAllocation(p1, p2) [48]	0.0 [49]	El score (0.0) sugiere una baja probabilidad de conexión basada en la fuerza y exclusividad de sus vecinos comunes[50].

5. Análisis Global y Recomendaciones

El análisis revela una red de carreteras **altamente conectada y balanceada** en términos de conectividad.

Hallazgos Principales

- **Madrid como Hub Central:** Domina en las métricas de **cercanía** e **intermediación**, haciéndola el nodo más crítico y estratégico para el flujo de tráfico en la red.
- **Conectividad Total:** La red no tiene nodos aislados (WCC: una componente única).

- **Estructura Cohesionada:** Existen grupos locales bien conectados, reflejado en altos coeficientes de *clustering* en ciertas ciudades.

Recomendación

Para mejorar la **resiliencia** de la red, se deben considerar conexiones alternativas que reduzcan la dependencia de **Madrid** como único *hub* principal, distribuyendo parte de su crucial rol de intermediación a otros nodos estratégicos como **Zaragoza**.