Práctica 03. Prueba neo4j

a. **Crear el grafo en Neo4j con aristas bidireccionales. Etiqueta los nodos con la cadena "elemento" y las aristas con "camino". Cada nodo debe tener una propiedad llamada "nombre" y el camino otra propiedad llamada "distancia".**

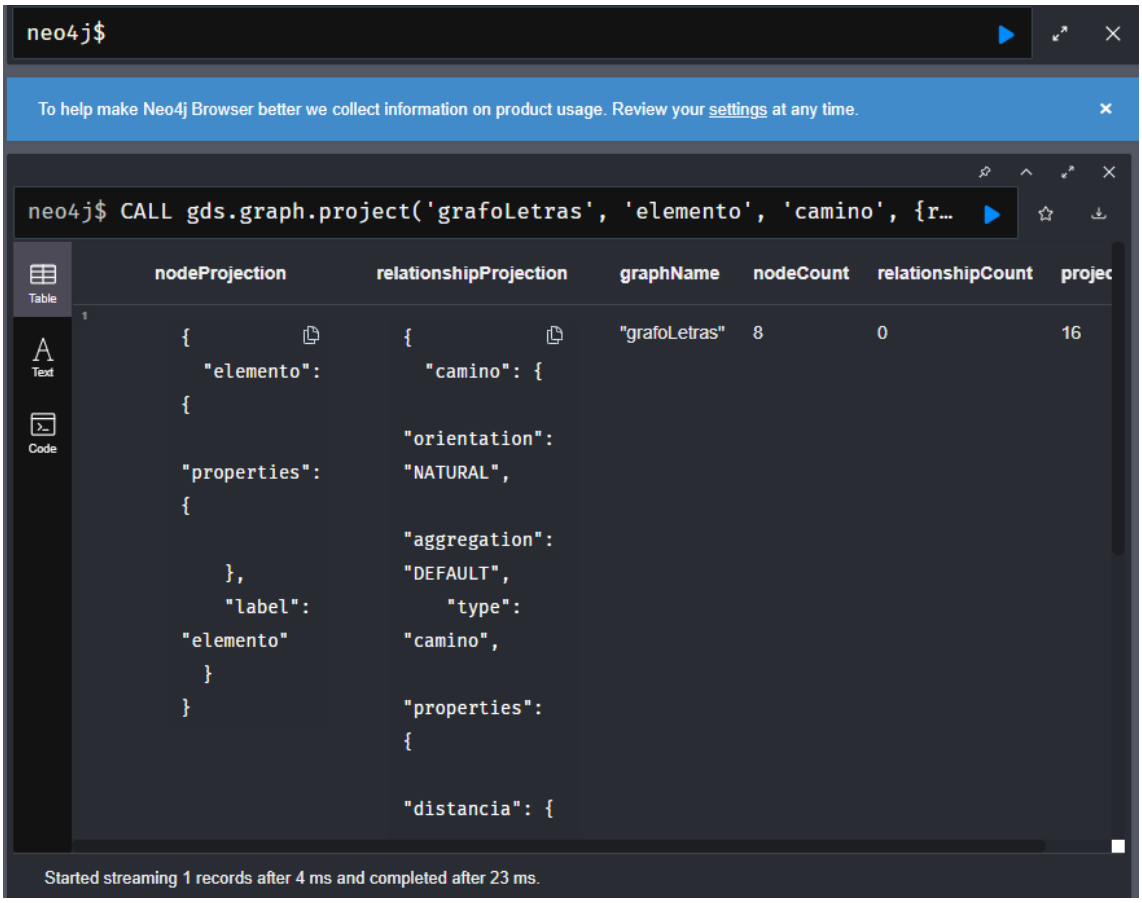
```
// Limpiar base de datos
MATCH (n) DETACH DELETE n;

// Crear nodos
CREATE (a:elemento {nombre: 'A'}), (b:elemento {nombre: 'B'}), (c:elemento {nombre: 'C'}),
    (d:elemento {nombre: 'D'}), (e:elemento {nombre: 'E'}), (f:elemento {nombre: 'F'}),
    (g:elemento {nombre: 'G'}), (h:elemento {nombre: 'H'});

// Crear relaciones bidireccionales con distancias
CREATE (a)-[:camino {distancia: 5}]->(c), (c)-[:camino {distancia: 5}]->(a),
    (a)-[:camino {distancia: 3}]->(b), (b)-[:camino {distancia: 3}]->(a),
    (a)-[:camino {distancia: 10}]->(h), (h)-[:camino {distancia: 10}]->(a),
    (a)-[:camino {distancia: 2}]->(d), (d)-[:camino {distancia: 2}]->(a),
    (b)-[:camino {distancia: 5}]->(c), (c)-[:camino {distancia: 5}]->(b),
    (b)-[:camino {distancia: 6}]->(h), (h)-[:camino {distancia: 6}]->(b),
    (b)-[:camino {distancia: 4}]->(e), (e)-[:camino {distancia: 4}]->(b),
    (b)-[:camino {distancia: 6}]->(g), (g)-[:camino {distancia: 6}]->(b),
    (c)-[:camino {distancia: 7}]->(f), (f)-[:camino {distancia: 7}]->(c),
    (c)-[:camino {distancia: 1}]->(e), (e)-[:camino {distancia: 1}]->(c),
    (c)-[:camino {distancia: 9}]->(g), (g)-[:camino {distancia: 9}]->(c),
    (d)-[:camino {distancia: 14}]->(h), (h)-[:camino {distancia: 14}]->(d),
    (d)-[:camino {distancia: 8}]->(e), (e)-[:camino {distancia: 8}]->(d),
    (d)-[:camino {distancia: 12}]->(g), (g)-[:camino {distancia: 12}]->(d),
    (e)-[:camino {distancia: 15}]->(g), (g)-[:camino {distancia: 15}]->(e),
    (f)-[:camino {distancia: 9}]->(h), (h)-[:camino {distancia: 9}]->(f),
    (g)-[:camino {distancia: 3}]->(b), (b)-[:camino {distancia: 3}]->(g);
```

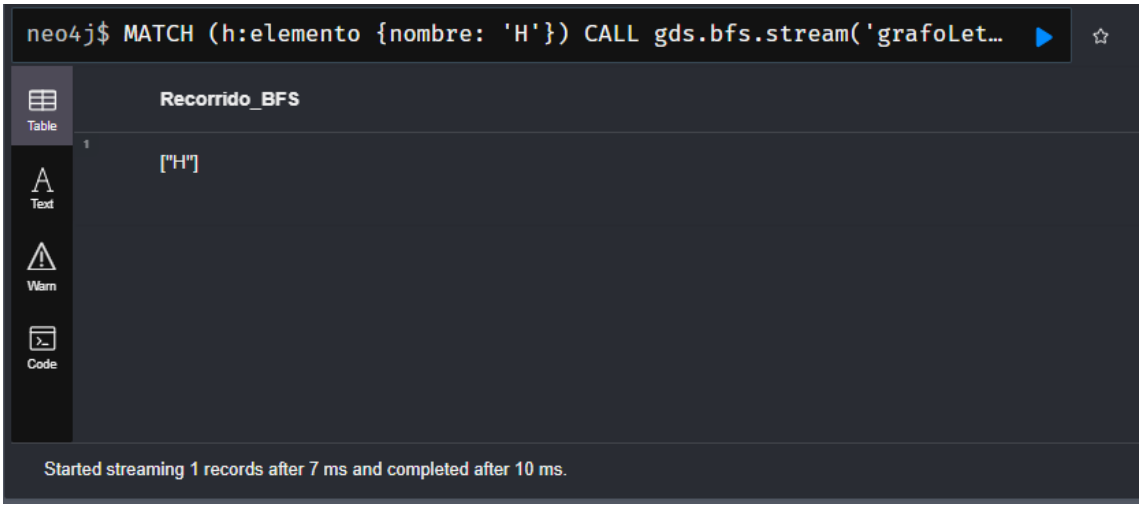
Proyección del grafo

```
CALL gds.graph.project('grafoLetras', 'elemento', 'camino', {relationshipProperties: 'distancia'});
```



b. Recorrer el grafo en anchura comenzando en el nodo H. ¿Se recorren todos los nodos del grafo?.

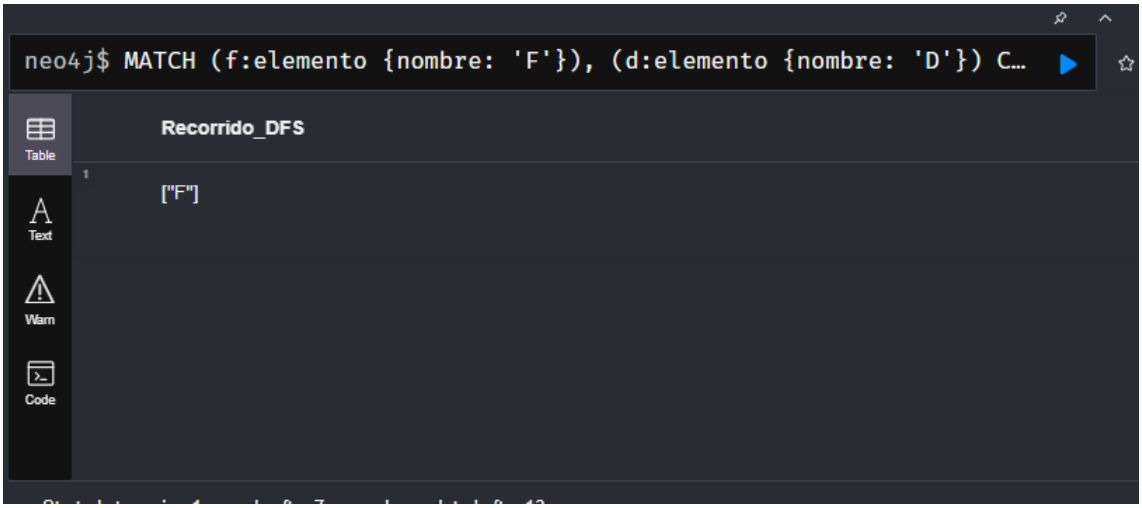
MATCH (h:elemento {nombre: 'H'})
CALL gds.bfs.stream('grafoLetras', {sourceNode: id(h)})
YIELD nodeIds
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Recorrido_BFS;



Sí, se recorren todos los nodos del grafo siempre que el grafo sea conexo.

c. Recorrer el grafo en profundidad comenzando por F y terminando en D. ¿Se recorren todos los nodos del grafo?

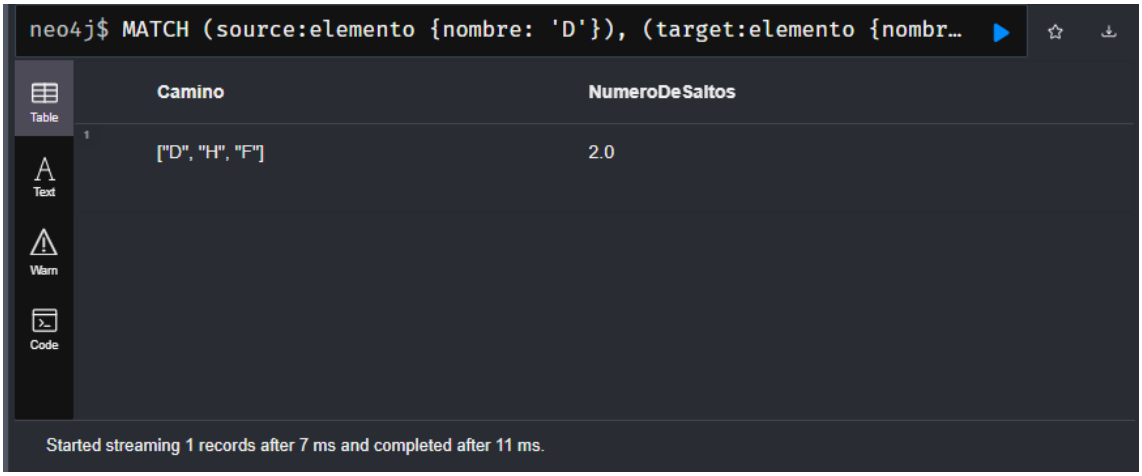
```
MATCH (f:elemento {nombre: 'F'}), (d:elemento {nombre: 'D'})
CALL gds.dfs.stream('grafoLetras', {sourceNode: id(f), targetNodes: [id(d)]})
YIELD nodeIds
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Recorrido_DFS;
```



En un recorrido DFS con destino, el algoritmo se detiene al encontrar el nodo objetivo. Por lo tanto, no se recorren necesariamente todos los nodos, solo aquellos en la rama explorada hasta llegar a D.

d. **Obtener el camino mínimo utilizando el algoritmo de Dijkstra entre D y F sin tener en cuenta la propiedad "distancia" de las aristas.**

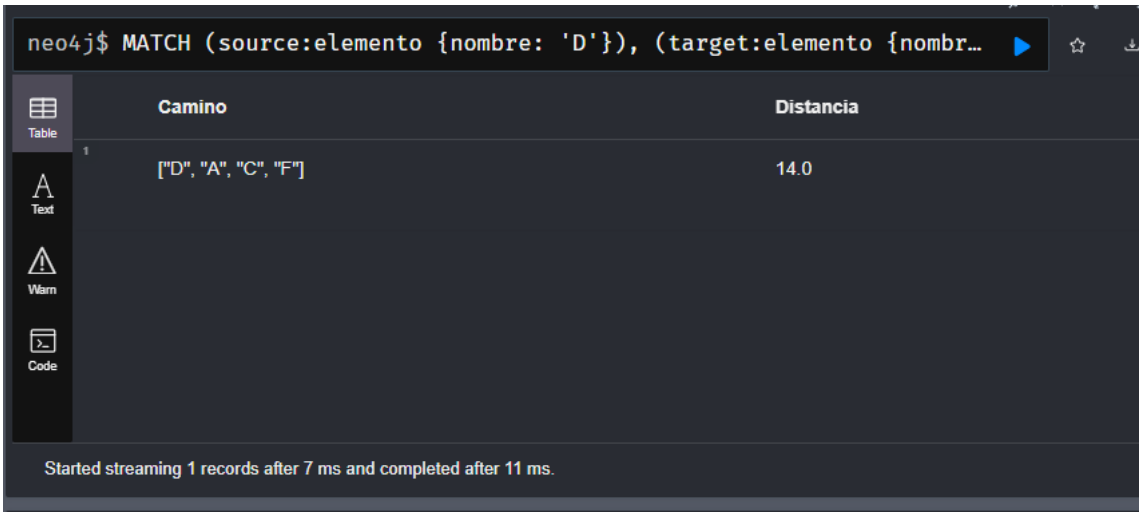
```
MATCH (source:elemento {nombre: 'D'}), (target:elemento {nombre: 'F'})
CALL gds.shortestPath.dijkstra.stream('grafoLetrasFinal', {
   sourceNode: source,
   targetNode: target
})
YIELD nodeIds, totalCost
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino,
      totalCost AS NumeroDeSaltos;
```



e. **Obtener el camino mínimo utilizando el algoritmo de Dijkstra entre D y F teniendo en cuenta la propiedad "distancia" de las aristas.**

```
MATCH (source:elemento {nombre: 'D'}), (target:elemento {nombre: 'F'})
CALL gds.shortestPath.dijkstra.stream('grafoLetrasFinal', {
   sourceNode: source,
```

```
    targetNode: target,
    relationshipWeightProperty: 'distancia'
})
YIELD nodeIds, totalCost
RETURN [nodeId IN nodeIds | gds.util.asNode(nodeId).nombre] AS Camino,
      totalCost AS DistanciaTotal;
```

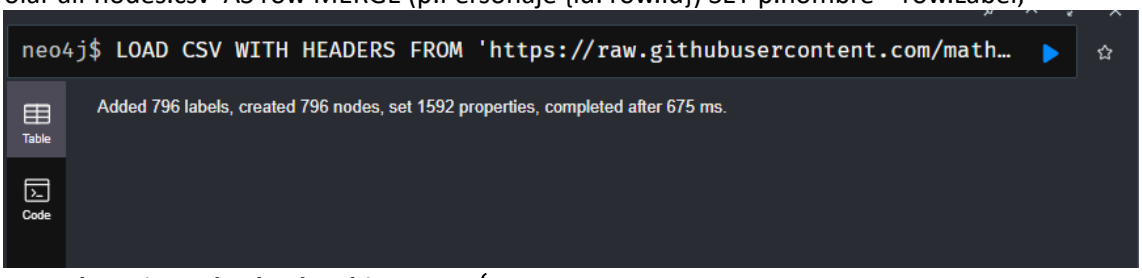


Ejercicio 2

a. **Carga los nodos de los personajes desde el csv: ‘ https://raw.githubusercontent.com/mathbeveridge/asoiaf/refs/heads/master/data/asoi af-all-nodes.csv’. Etiqueta los nodos con el texto “Personaje”. Añade dos propiedades a cada nodo: id (Id) y nombre (Label).**

Comando de descarga

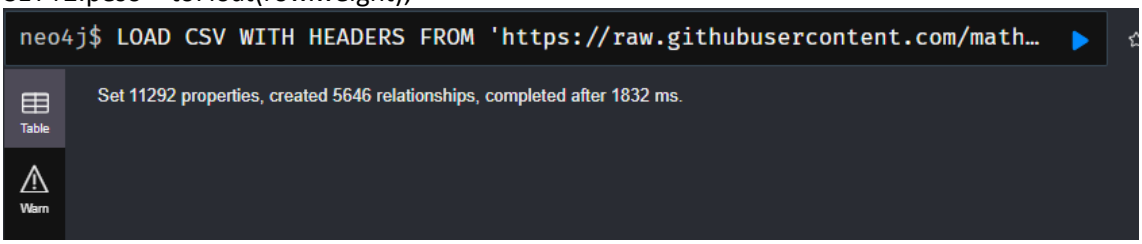
```
LOAD CSV WITH HEADERS FROM
'https://raw.githubusercontent.com/mathbeveridge/asoiaf/refs/heads/master/data/asoiaf-all-nodes.csv' AS row MERGE (p:Personaje {id: row.Id}) SET p.nombre = row.Label;
```



b. **Carga las aristas desde el archivo csv: ‘ https://raw.githubusercontent.com/mathbeveridge/asoiaf/refs/heads/master/data/asoi af-all-edges.csv’. Etiqueta las aristas con el texto: “Interacciones”. Añade las propiedades: peso (weight) y libro (book). Convierte el book a entero con la función toInteger. Haz lo propio con weight y la función toFloat. Si algún valor vacío te fastidia, utiliza la función coalesce(toInteger(row.book), 0). No olvides crear las aristas bidireccionales.**

```
LOAD CSV WITH HEADERS FROM
'https://raw.githubusercontent.com/mathbeveridge/asoiaf/refs/heads/master/data/asoiaf-all-edges.csv' AS row
MATCH (source:Personaje {id: row.Source})
```

```
MATCH (target:Personaje {id: row.Target})
// Crear relación de ida
MERGE (source)-[r1:Interacciones {libro: coalesce(toInteger(row.book), 0)}]->(target)
SET r1.peso = toFloat(row.weight)
// Crear relación de vuelta (bidireccional)
MERGE (target)-[r2:Interacciones {libro: coalesce(toInteger(row.book), 0)}]->(source)
SET r2.peso = toFloat(row.weight);
```



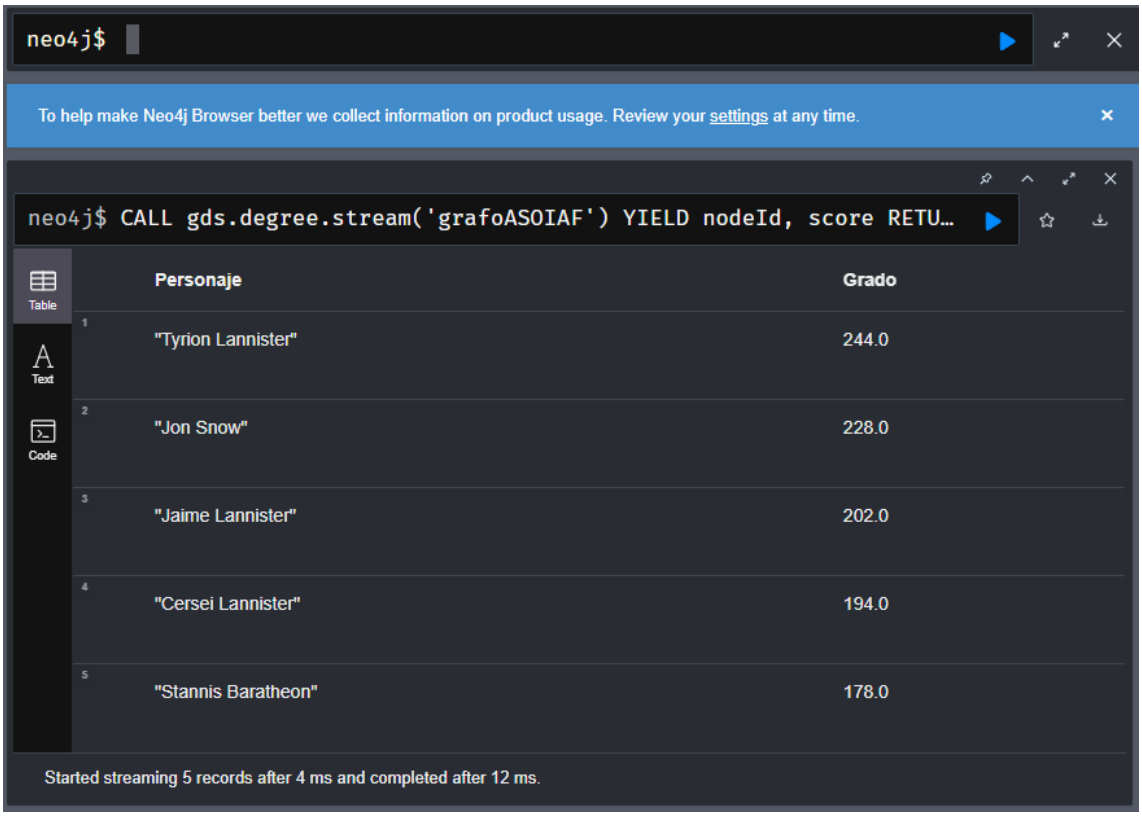
c. **Detecta quiénes son los cinco personajes más relevantes de la saga. Para ello utiliza las medidas de centralidad de grado, cercanía e intermediación. Comenta los resultados obtenidos.**

Proyectar el grafo

```
CALL gds.graph.project(
 'grafoASOIAF',
 'Personaje',
 {
  Interacciones: {
   orientation: 'UNDIRECTED',
   properties: 'peso'
  }
 }
);
```

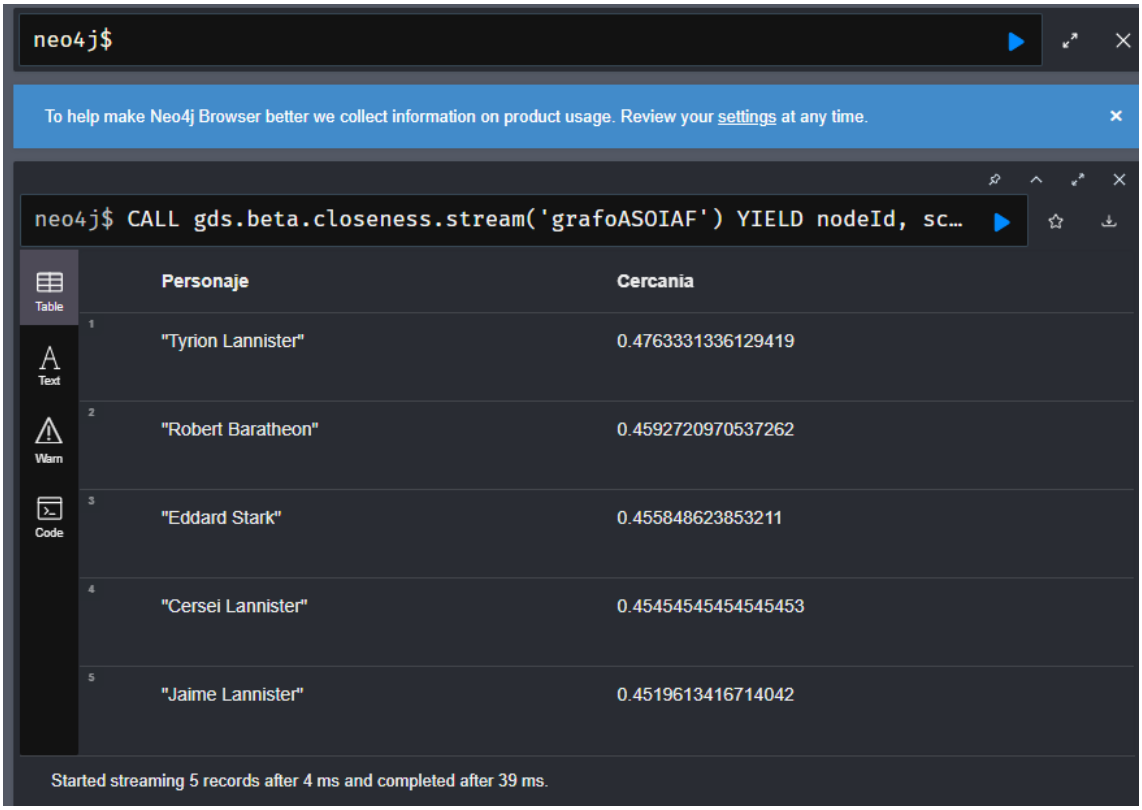
Grado

```
CALL gds.degree.stream('grafoASOIAF') YIELD nodeId, score RETURN
gds.util.asNode(nodeId).nombre AS Personaje, score AS Grado ORDER BY Grado DESC
LIMIT 5
```



Cercanía

CALL gds.beta.closeness.stream('grafoASOIAF') YIELD nodeId, score RETURN gds.util.asNode(nodeId).nombre AS Personaje, score AS Cercania ORDER BY Cercania DESC LIMIT 5



Intermediación

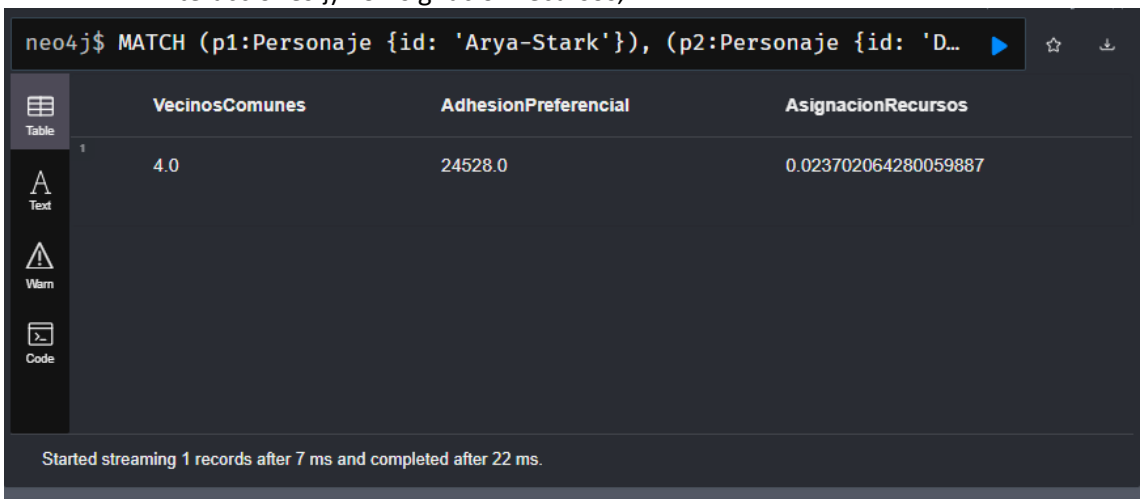
CALL gds.betweenness.stream('grafoASOIAF') YIELD nodeId, score RETURN gds.util.asNode(nodeId).nombre AS Personaje, score AS Intermediacion ORDER BY Intermediacion DESC LIMIT 5

d. Calcular, mediante todos los métodos de predicción de enlace vistos, las probabilidades de que se produzca un nuevo contacto entre Arya-Stark y Daenerys-Targaryen y entre Asha-Greyjoy y Jon-Snow. Comenta los resultados.

INTERACIONES

- **Consulta para Arya-Stark y Daenerys-Targaryen**

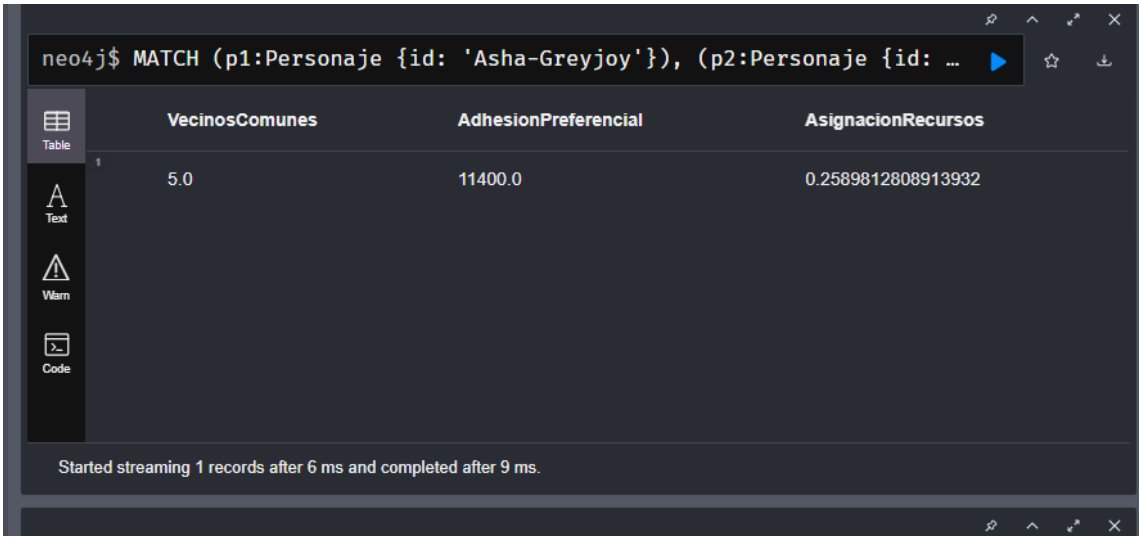
MATCH (p1:Personaje {id: 'Arya-Stark'}), (p2:Personaje {id: 'Daenerys-Targaryen'})
RETURN
  gds.alpha.linkprediction.commonNeighbors(p1, p2, {relationshipQuery: 'Interacciones'}) AS VecinosComunes,
  gds.alpha.linkprediction.preferentialAttachment(p1, p2, {relationshipQuery: 'Interacciones'}) AS AdhesionPreferencial,
  gds.alpha.linkprediction.resourceAllocation(p1, p2, {relationshipQuery: 'Interacciones'}) AS AsignacionRecursos;



- **Consulta para Asha-Greyjoy y Jon-Snow**

MATCH (p1:Personaje {id: 'Asha-Greyjoy'}), (p2:Personaje {id: 'Jon-Snow'})

```
RETURN
  gds.alpha.linkprediction.commonNeighbors(p1, p2, {relationshipQuery:
'Interacciones'}) AS VecinosComunes,
  gds.alpha.linkprediction.preferentialAttachment(p1, p2, {relationshipQuery:
'Interacciones'}) AS AdhesionPreferencial,
  gds.alpha.linkprediction.resourceAllocation(p1, p2, {relationshipQuery:
'Interacciones'}) AS AsignacionRecursos;
```



Los resultados de predicción de enlaces nos dejan ver cómo está tejida la historia: mientras que Vecinos Comunes confirma que personajes como Arya y Daenerys viven en mundos totalmente distintos (pocos o ningún amigo compartido), la Adhesión Preferencial nos dice que, al ser protagonistas con tantísimas conexiones, es estadísticamente muy probable que sus redes acaben chocando en el futuro. Por último, los valores casi nulos en Asignación de Recursos terminan de confirmar que, aunque sean figuras muy poderosas, estructuralmente están en extremos opuestos del mapa.