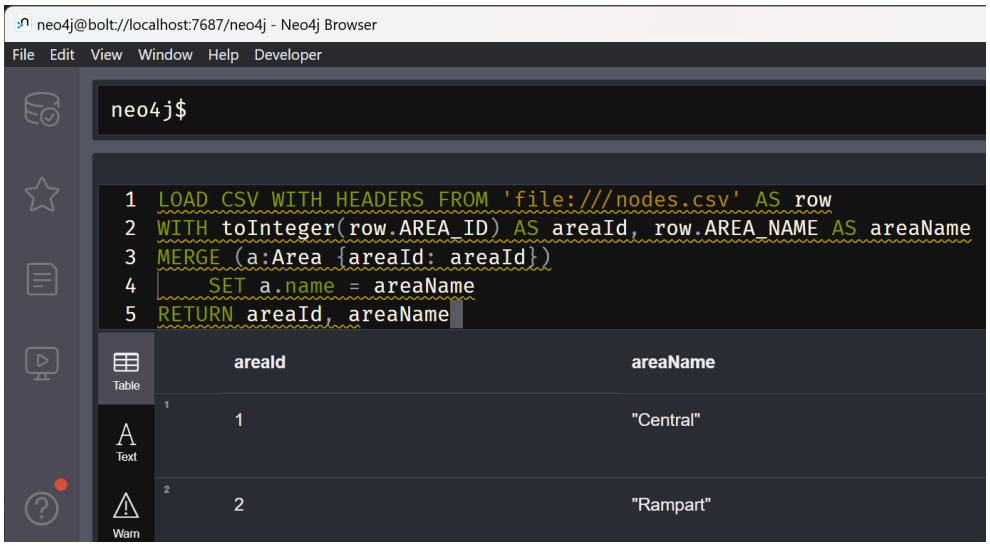
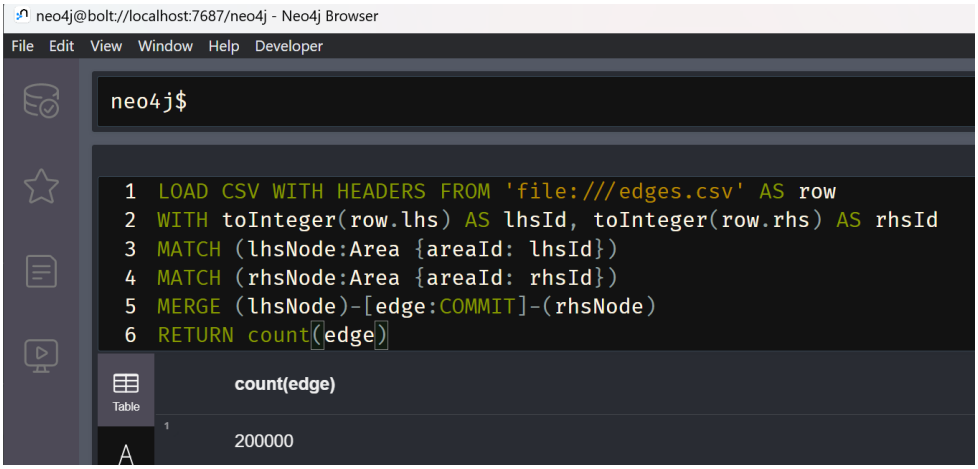
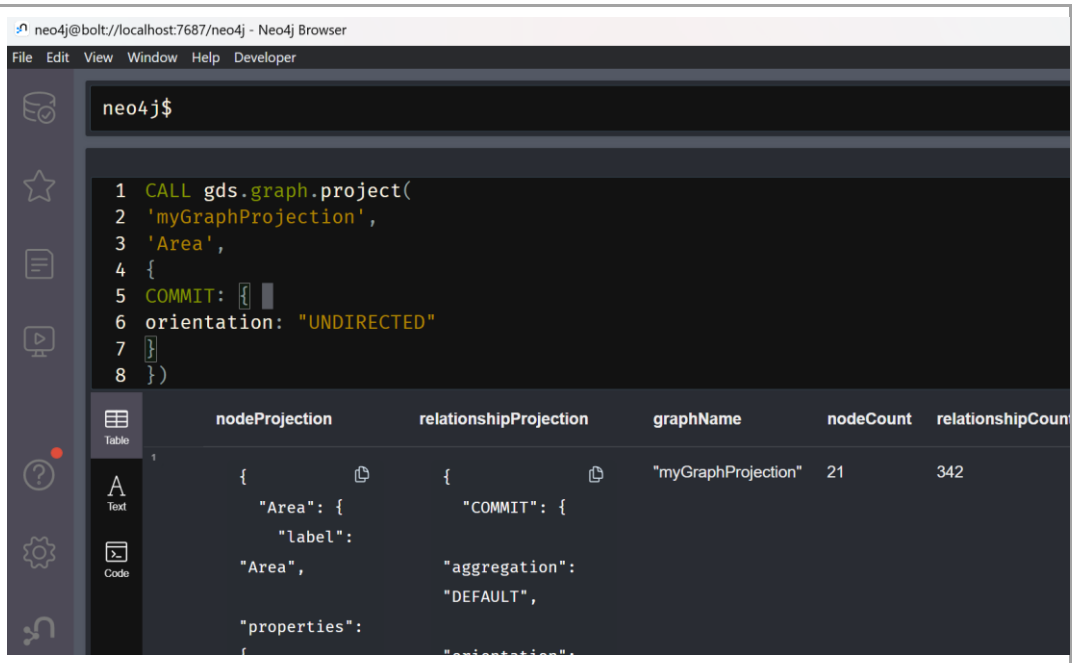
Export für Neo4j	select t1.area_id, t1.crime_code, t2.area_id, t2.crime_code from crime t1, crime t2 where t1.crime_code = t2.crime_code and t1.area_id > t2.area_id call CSVWRITE ('C:/users/beni/downloads/edges.csv', 'select top 200000 t1.area_id, t2.area_id from crime t1, crime t2 where t1.crime_code = t2.crime_code and t1.area_id > t2.area_id'); call CSVWRITE ('C:/users/beni/downloads/nodes.csv', 'select area_id, area_name from area');
Neo4j import csv (Nodes)	LOAD CSV WITH HEADERS FROM 'file:///nodes.csv' AS row WITH toInteger(row.AREA_ID) AS areaId, row.AREA_NAME AS areaName MERGE (a:Area {areaId: areaId}) SET a.name = areaName RETURN areaId, areaName 
Neo4j import csv (Edges)	LOAD CSV WITH HEADERS FROM 'file:///edges.csv' AS row WITH toInteger(row.lhs) AS lhsId, toInteger(row.rhs) AS rhsId MATCH (lhsNode:Area {areaId: lhsId})

	MATCH (rhsNode:Area {areaId: rhsId}) MERGE (lhsNode)-[edge:COMMIT]-(rhsNode) RETURN count(edge) 

Neo4j create graph (gds)	CALL gds.graph.project(
	'myGraphProjection',
	'Area',
	{
	COMMIT: {
	orientation: "NATURAL"
	}
	})



Neo4j calculate centrality

```
CALL gds.degree.stream(
"myGraphProjection"
) YIELD
nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score AS degree
ORDER BY degree DESC, name DESC;
```