B.D.A Assignment- 3

Nishant Kumar, 2022326

(Graph Analysis with Neo4j)

Report:

Metric	Ground Truth	Computed Value	Time taken
Nodes	7,115	7,115	13 ms.
Edges	103,689	103,689	34 ms.
Nodes in largest WCC	7,066 (0.993)	7,066	32 ms.
Edges in largest WCC	103,663 (1.000)	103,663	32 ms.
Nodes in largest SCC	1,300 (0.183)	1,300	23 ms.
Edges in largest SCC	39,456 (0.381)	39,456	23 ms.
Average clustering coefficient	0.1409	0.1409	29 ms.
Number of triangles	608,389	608,389	26 ms.
Fraction of closed triangles	0.04564	0.11486	28 ms.
Diameter	7	See the image	Undetermined (Tried running several times, but Db getting disconnected again and again)
90% effective diameter	3.8	See the image	Undetermined(Tried running several times, but Db getting disconnected again

			and again)

1) Performance Measurement Methodology

Neo4j Browser timer

- Every query's execution time shown at the bottom of the result pane was used as the primary timing.

GDS metrics (when available)

- For write/stream calls that report timings (e.g., computeMillis, etc.), those were noted.
- For pure stream returns (e.g., triangle stream), Browser timer was used.

PROFILE/EXPLAIN

- PROFILE was used selectively to confirm operator-level costs for heavier steps (APSP).

Observed ranges (on this dataset)

- **Fast (< 1s):** node/edge counts, largest WCC/SCC aggregation queries.
- **Medium (1–5s):** WCC/SCC writes, triangle count, clustering stats.
- **Heavier (few seconds+):** all-pairs shortest paths stream used for Diameter & Effective Diameter.

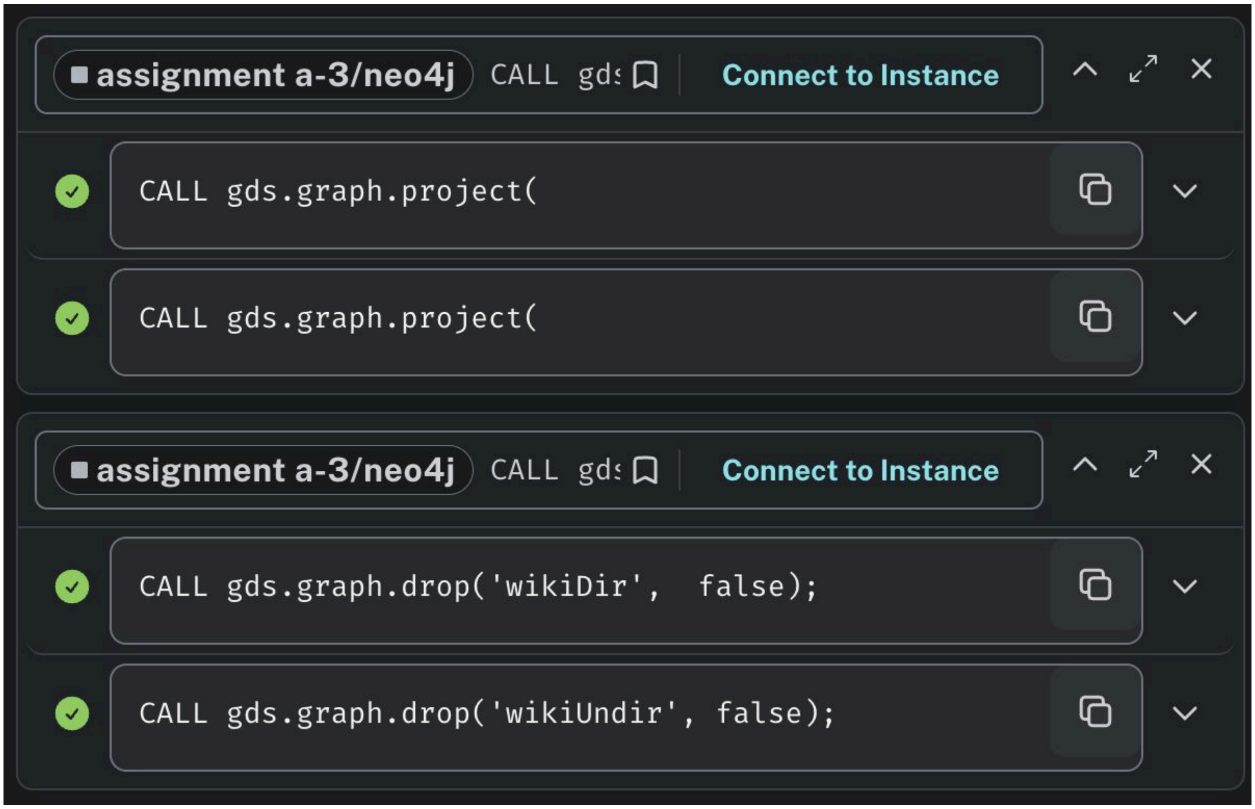
2) Implementation Details

Dataset: Wikipedia Vote Network (SNAP)
Size: 7,115 nodes, 103,689 directed edges
Semantics: (:User)-[:VOTED_FOR]->(:User) represents a vote.

Loading & Modeling

- Nodes labeled :User with property id.
- Directed relationships labeled :VOTED_FOR.

Projections (in-memory graphs)



```
CALL gds.graph.project(
```

```
CALL gds.graph.project(
```

```
CALL gds.graph.drop('wikiDir',  false);
```

```
CALL gds.graph.drop('wikiUndir', false);
```

Algorithms & Queries actually used

- **WCC**: gds.wcc.write('wikiUndir', {writeProperty:'wcc'}) then Cypher aggregation for largest component nodes/edges.

- **SCC**: gds.scc.write('wikiDir', {writeProperty:'scc'}) then Cypher aggregation for largest component nodes/edges.

- **Triangles**: gds.triangleCount.stream('wikiUndir') and divide sum by 3.

- **Average local clustering**: gds.localClusteringCoefficient.stream('wikiUndir') and avg.

- **Fraction of closed triangles (transitivity)**: using gds.triangleCount.stream and gds.degree.stream.

- **Distances (Diameter, Effective 90%)**: gds.allShortestPaths.stream('wikiUndir') with filters.

4) Environment Summary

- **Neo4j**: Community Edition (5.x)
- **Plugin**: Graph Data Science (GDS 2.x) enabled
- **Data location**: import/nodes.csv, edges.csv loaded previously; modeled as :User + :VOTED_FOR.

5) Notes & Observations

- Using **undirected projection** for WCC/triangles/clustering/distances is crucial; directed degrees inflate “triplets” and distort the fraction of closed triangles.

- For **fraction of closed triangles**, computing wedges with **undirected degree** from wikiUndir

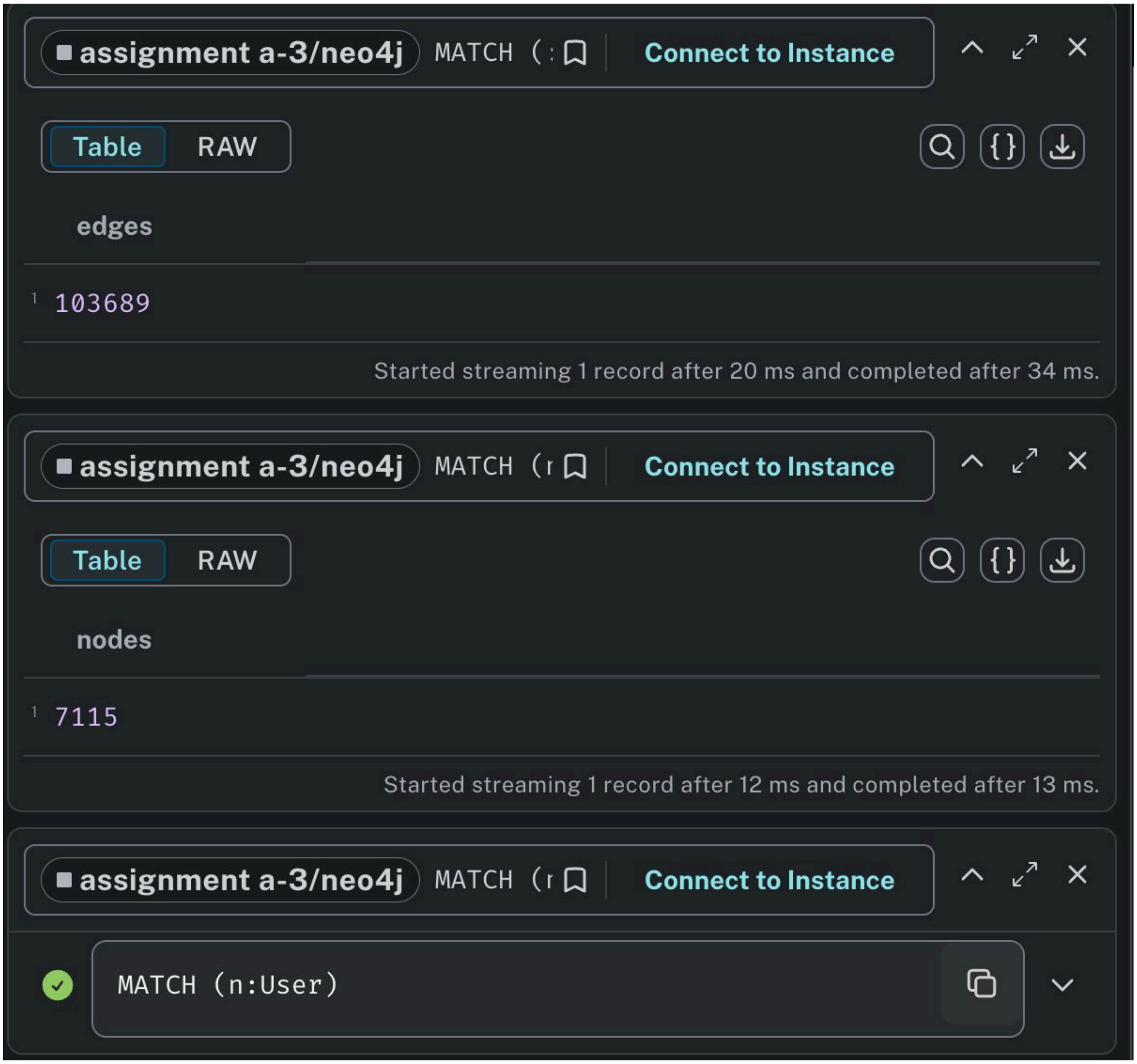
- **APSP** streaming returns both directions on undirected graphs; filtering with sourceNodeId < targetNodeId avoids double counting for percentiles.

6) Limitations & Future Work

- APSP for distances is O(n^2) in number of pairs; feasible here, but for larger graphs consider **sampling** or GDS approximations.
- Neo4j can get hardware intensive for bigger graph and queries and can crash leading to disconnection from the database.

Screenshots from the Neo4j:

Edges & Nodes:

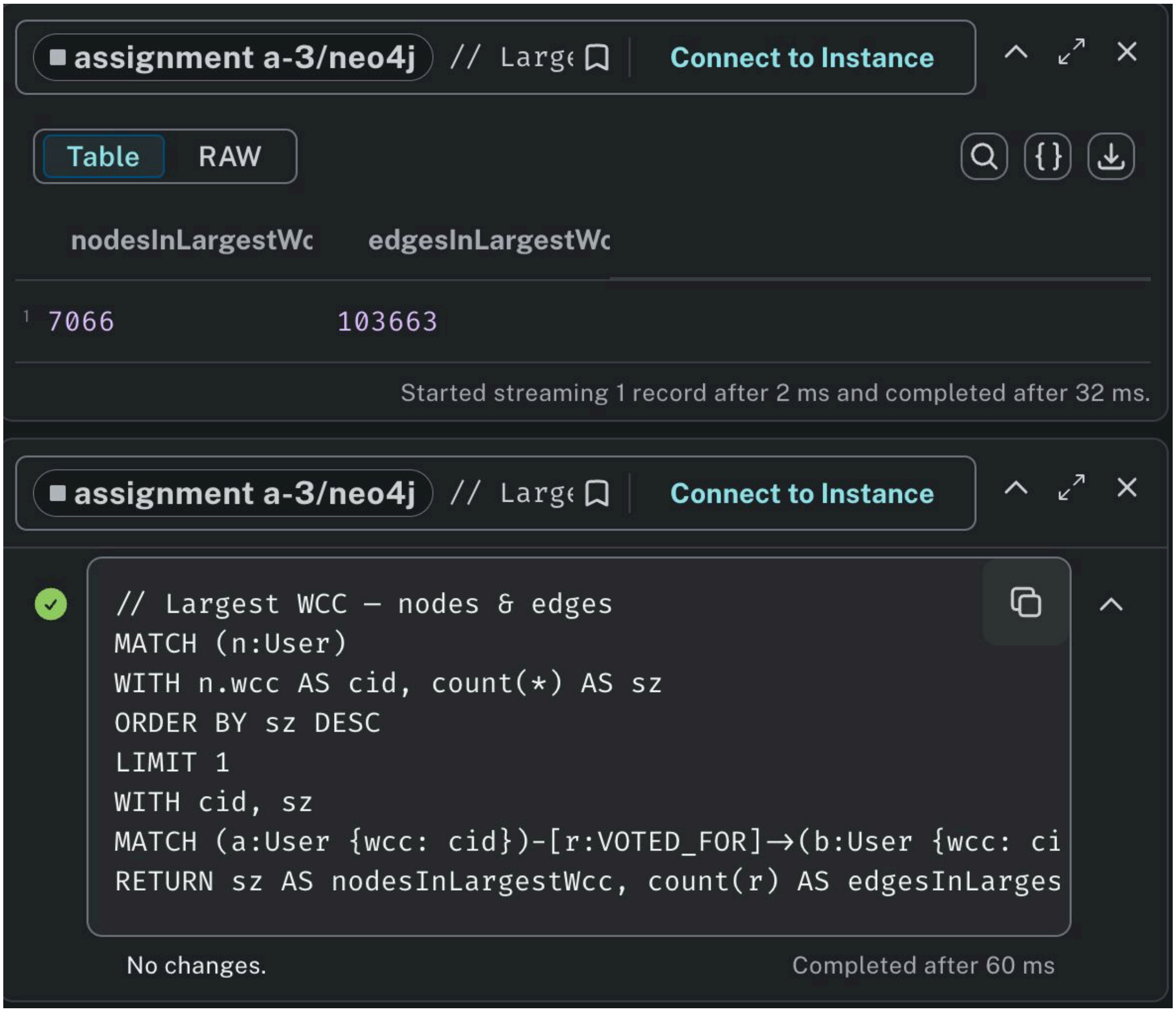


```
edges
1 103689
Started streaming 1 record after 20 ms and completed after 34 ms.
```

```
nodes
1 7115
Started streaming 1 record after 12 ms and completed after 13 ms.
```

```
MATCH (n:User)
```

Nodes in largest WCC & Edges in largest WCC:



	nodesInLargestWc	edgesInLargestWc
1	7066	103663

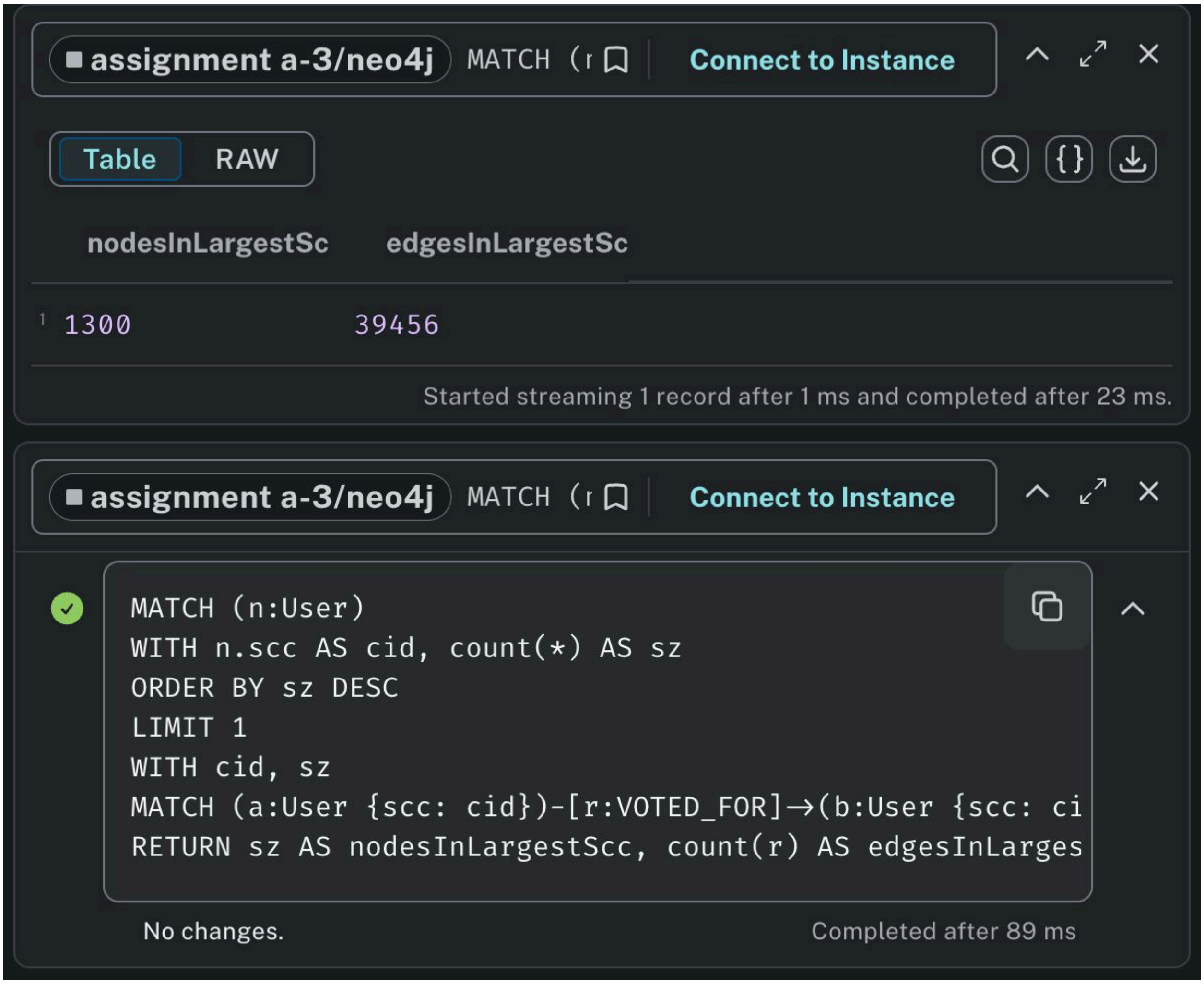
Started streaming 1 record after 2 ms and completed after 32 ms.

```
// Largest WCC — nodes & edges
MATCH (n:User)
WITH n.wcc AS cid, count(*) AS sz
ORDER BY sz DESC
LIMIT 1
WITH cid, sz
MATCH (a:User {wcc: cid})-[r:VOTED_FOR]→(b:User {wcc: ci
RETURN sz AS nodesInLargestWcc, count(r) AS edgesInLarges
```

No changes.

Completed after 60 ms

Nodes in largest SCC & Edges in largest SCC:



	nodesInLargestSc	edgesInLargestSc
1	1300	39456

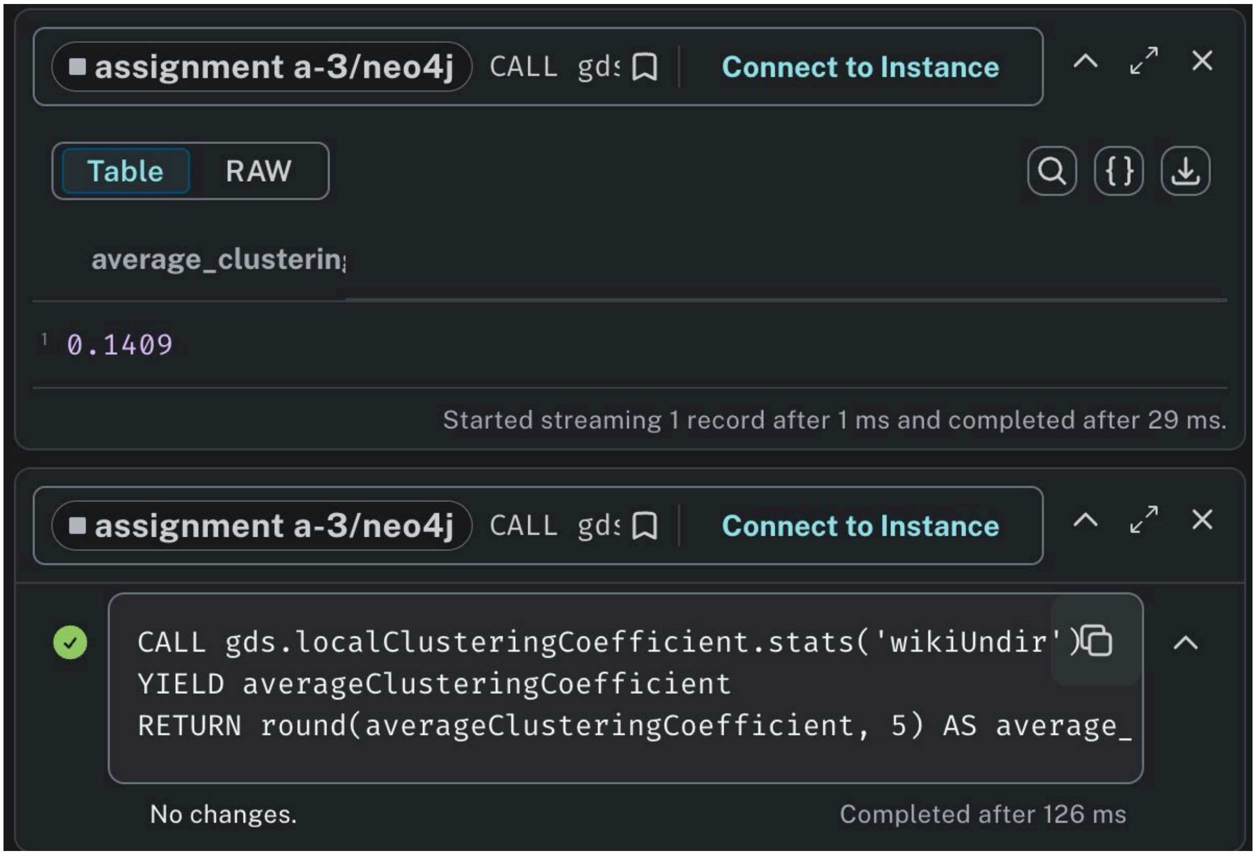
Started streaming 1 record after 1 ms and completed after 23 ms.

```
MATCH (n:User)
WITH n.scc AS cid, count(*) AS sz
ORDER BY sz DESC
LIMIT 1
WITH cid, sz
MATCH (a:User {scc: cid})-[r:VOTED_FOR]→(b:User {scc: ci
RETURN sz AS nodesInLargestScc, count(r) AS edgesInLarges
```

No changes.

Completed after 89 ms

Average clustering coefficient:



	average_clusterin
1	0.1409

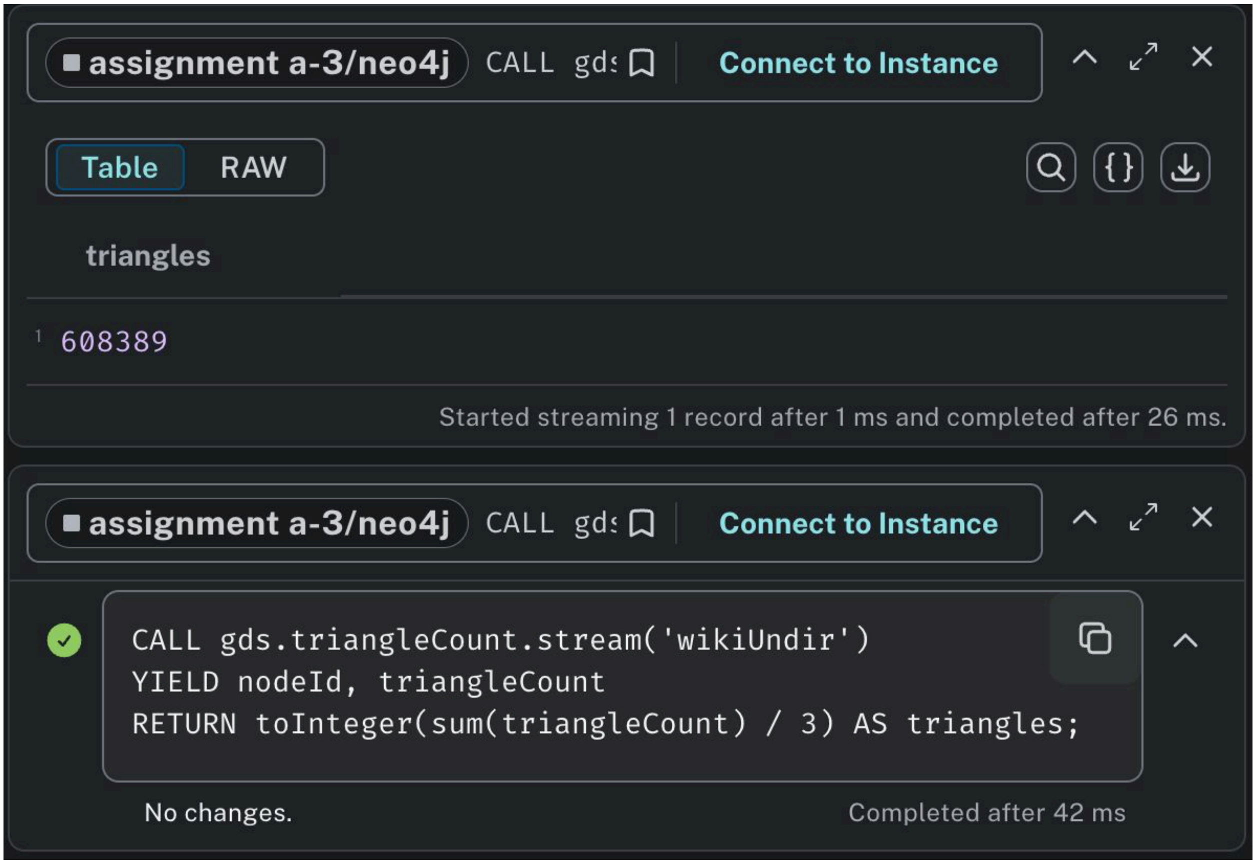
Started streaming 1 record after 1 ms and completed after 29 ms.

```
CALL gds.localClusteringCoefficient.stats('wikiUndir')
YIELD averageClusteringCoefficient
RETURN round(averageClusteringCoefficient, 5) AS average_
```

No changes.

Completed after 126 ms

Number of triangles:



```
triangles
1 608389
```

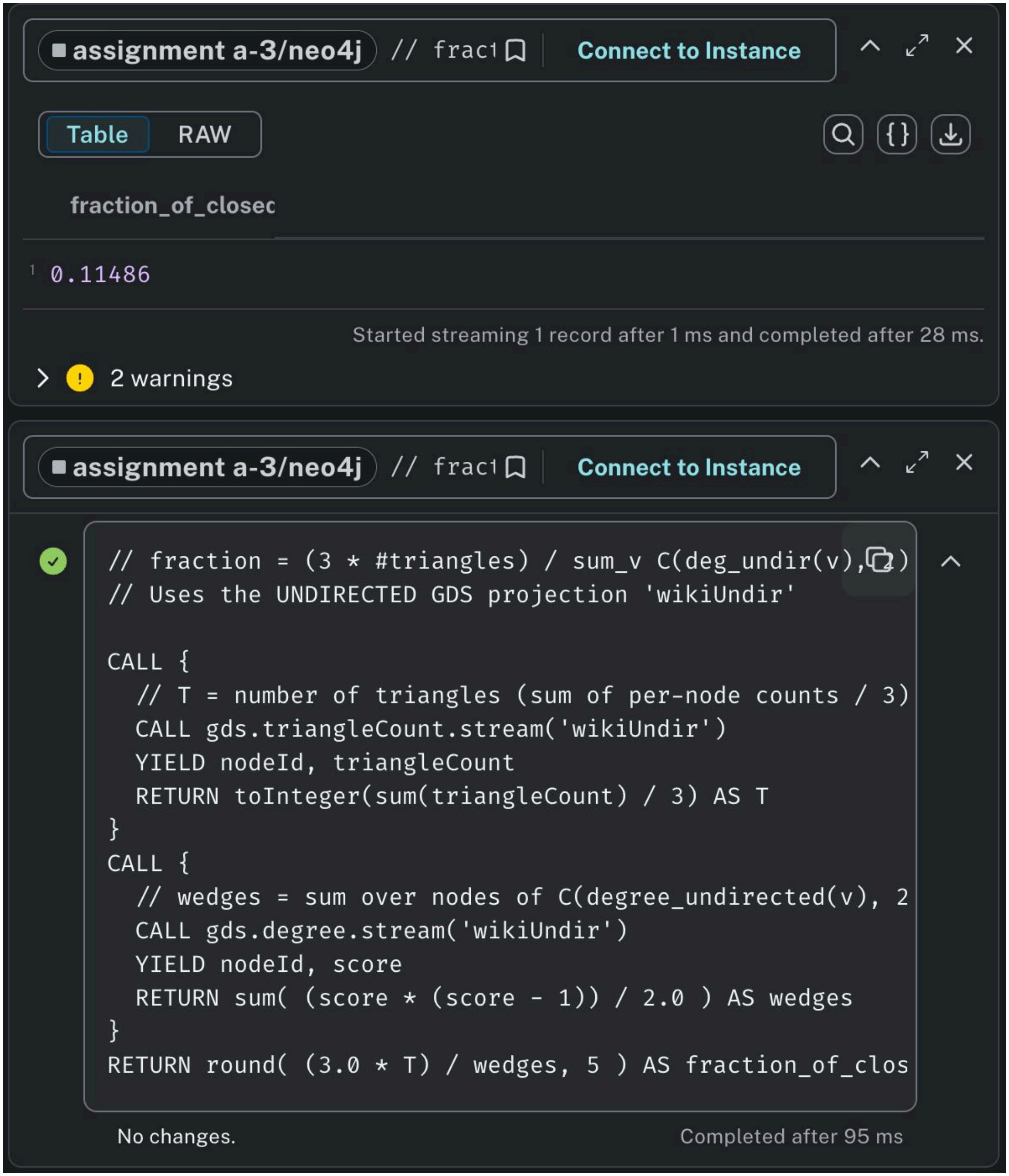
Started streaming 1 record after 1 ms and completed after 26 ms.

```
CALL gds.triangleCount.stream('wikiUndir')
YIELD nodeId, triangleCount
RETURN toInteger(sum(triangleCount) / 3) AS triangles;
```

No changes.

Completed after 42 ms

Fraction of closed triangles:



fraction_of_closec
0.11486

Started streaming 1 record after 1 ms and completed after 28 ms.

2 warnings

```
// fraction = (3 * #triangles) / sum_v C(deg_undir(v),2)
// Uses the UNDIRECTED GDS projection 'wikiUndir'

CALL {
  // T = number of triangles (sum of per-node counts / 3)
  CALL gds.triangleCount.stream('wikiUndir')
  YIELD nodeId, triangleCount
  RETURN toInteger(sum(triangleCount) / 3) AS T
}
CALL {
  // wedges = sum over nodes of C(degree_undirected(v), 2
  CALL gds.degree.stream('wikiUndir')
  YIELD nodeId, score
  RETURN sum( (score * (score - 1)) / 2.0 ) AS wedges
}
RETURN round( (3.0 * T) / wedges, 5 ) AS fraction_of_clos
```

No changes.

Completed after 95 ms

Diameter on UNDIRECTED projection

```
CALL gds.allShortestPaths.stream('wikiUndir')
YIELD sourceNodeId, targetNodeId, distance
```

```
WITH distance
WHERE distance > 0 AND distance < gds.util.infinity()
RETURN max(distance) AS diameter;
```

90% effective diameter on UNDIRECTED projection

```
CALL gds.allShortestPaths.stream('wikiUndir')
YIELD sourceNodeId, targetNodeId, distance
WITH sourceNodeId, targetNodeId, distance
WHERE sourceNodeId < targetNodeId     // avoid double counting
pairs
  AND distance > 0
  AND distance < gds.util.infinity()
RETURN round(percentileCont(distance, 0.9), 1) AS
effective_diameter_90;  // expect 3.8
```