



Adventist University of Central Africa

P.O. Box 2461 Kigali, Rwanda | www.auca.ac.rw | info@auca.ac.rw

|

Graph Analytics of the USA Roads Network

Analytics using Neo4j, Graph Data Science (GDS), Python and Plotly

Course Name: Big Data Analytics

Course Code: MSDA9215

Lecturer: Temitope Oguntade

Date: January 21, 2026

Project Team:

Names	Student ID Number
RUSINGIZWA Jean Pierre	101086
UWASE Carine	101030
AKIRI Olivier	101062

Academic Year: 2025–2026

Contents

1	Task 1: Total Number of Intersections and Roads	4
2	Task 2: Shortest Path Between Two Intersections	5
3	Task 3: Intersections with Degree Greater Than a Threshold	7
4	Task 4: Betweenness Centrality Analysis	9
5	Task 5: Dashboard and Key Metrics using Plotly	10
6	Task 6: Degree Distribution	11
7	Task 7: Top 10 Most Connected Intersections	12
8	Task 8: Intersection Categories by Degree	14
9	Task 9: Degree Distribution	16

Abstract

This report presents graph-based analytics on a United States roads network dataset using Neo4j and its specialized libraries like APOC and Graph Data Science (GDS). Euclidean distance is used as the edge weight to enable shortest path and centrality analysis. We load intersections as nodes and roads as relationships, compute graph statistics, shortest paths, degree-based queries, and centrality measures. A Plotly dashboard is generated to visualize key metrics including counts, degree distribution, and top connected intersections.

Dataset Overview

The dataset represents a planar roads network (graph) where:

- Vertices = intersections (nodes)
- Edges = roads (relationships)
- Edge weights = Euclidean distances between endpoints

The full dataset contains **87,575 intersections** and **121,961 roads**. It is sparse with an average degree around **2.8** which is typical for real-world road networks.

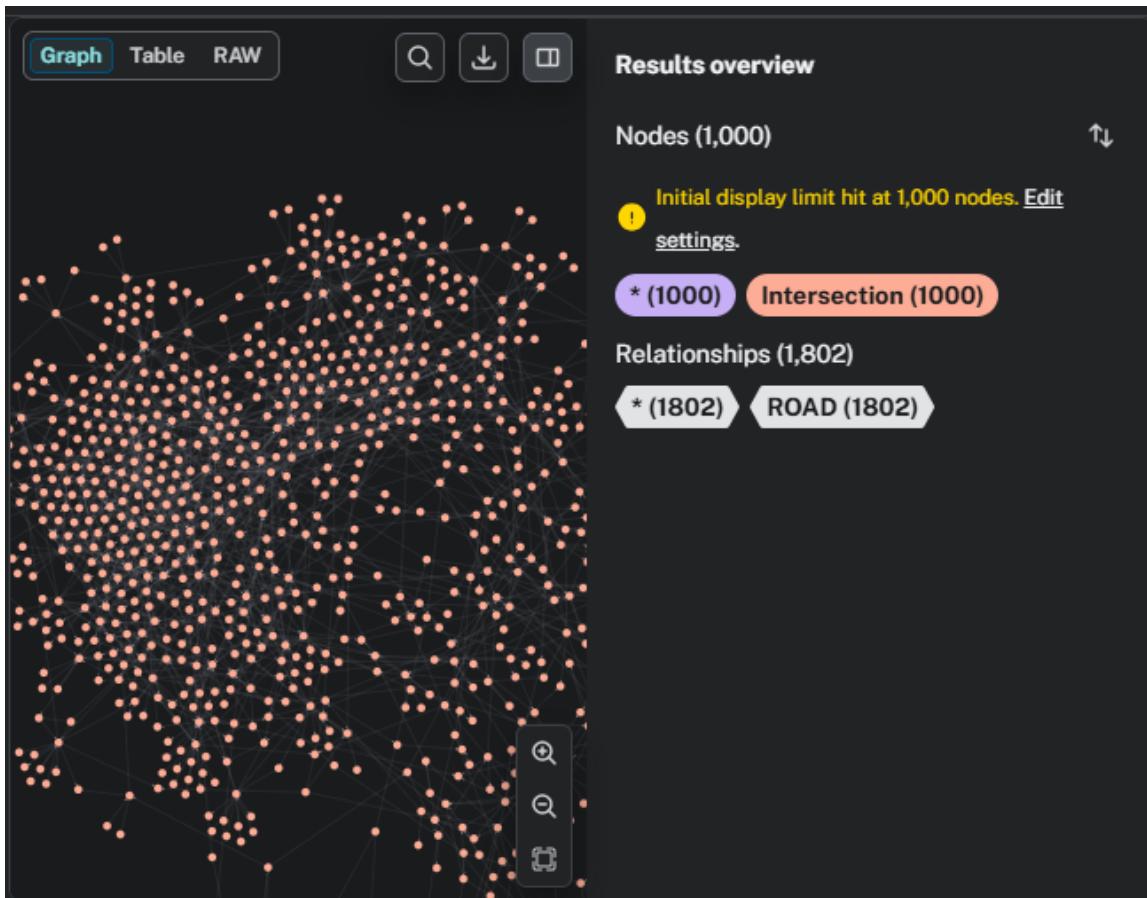


Figure 1: Roads and junctionsnodes

Data Modeling in Neo4j

Each intersection is represented as a node labeled `Intersection` with properties including a unique identifier and spatial coordinates. Roads are modeled as `ROAD` relationships connecting intersections. Duplicate directional relationships were removed to ensure each road is counted once.

Node label: `:Intersection` with properties `id`, `x`, `y`

Relationship: `:ROAD` connecting intersections, with property `distance` (from, to)

Recommended Constraint

```

1 CREATE CONSTRAINT intersection_id_unique IF NOT EXISTS
2 FOR (i:Intersection) REQUIRE i.id IS UNIQUE;
```

Dataset Upload Queries

Upload intersections.csv

```
1 LOAD CSV WITH HEADERS FROM 'file:///intersections.csv' AS row
2 CREATE (:Intersection {
3   id: toInteger(row.id),
4   x: toInteger(row.x),
5   y: toInteger(row.y)
6});
```

Upload roads.csv

```
1 LOAD CSV WITH HEADERS FROM 'file:///roads.csv' AS row
2 MATCH (a:Intersection {id: toInteger(row.from)}),
3       (b:Intersection {id: toInteger(row.to)})
4 CREATE (a)-[:ROAD {distance: toFloat(row.distance)}]->(b);
```

1. Task 1: Total Number of Intersections and Roads

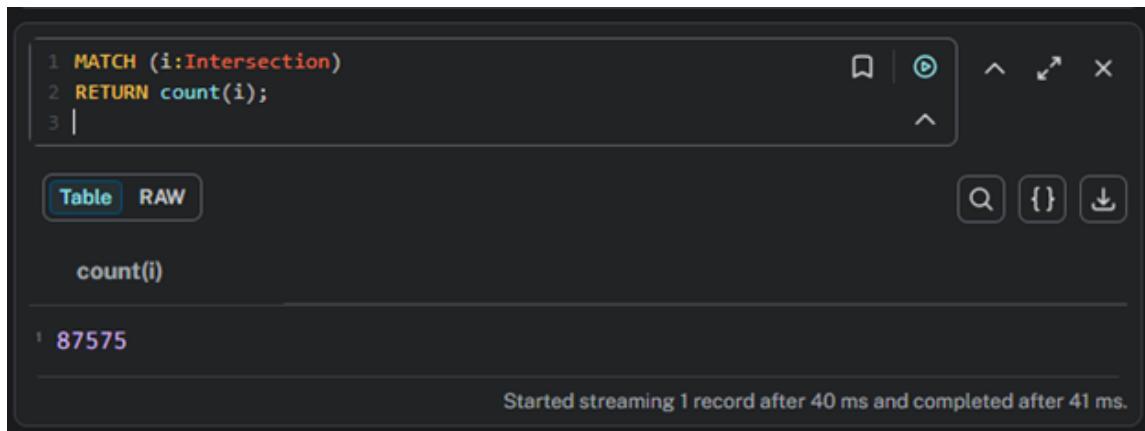
Query

```
1 MATCH (n:Intersection)
2 RETURN count(n) AS total_intersections;
3
4 MATCH ()-[r:ROAD]->()
5 RETURN count(r) AS total_roads;
```

Explanation

The first query counts only nodes labeled `Intersection`. The second query counts only relationships of type `ROAD`.

Output Screenshots



A screenshot of the Neo4j browser interface. At the top, there is a code editor window containing the following Cypher query:

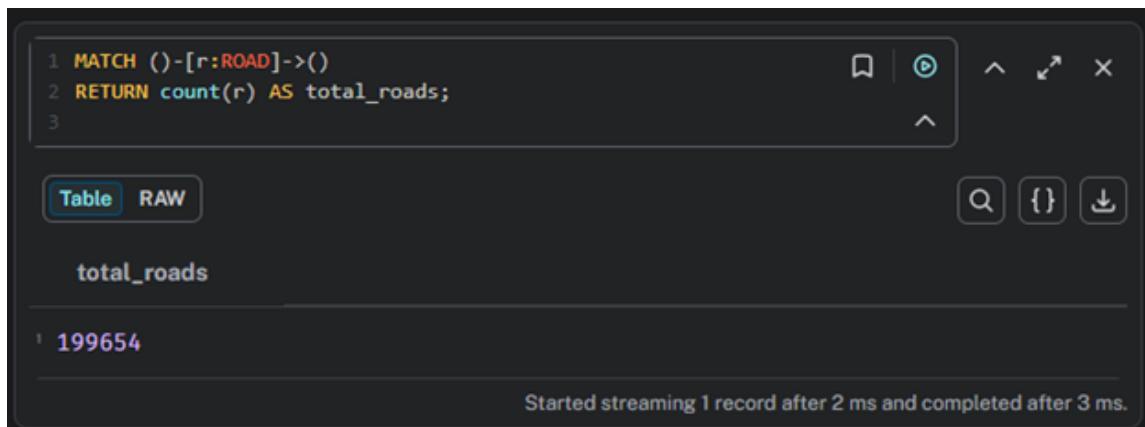
```
1 MATCH (i:Intersection)
2 RETURN count(i);
```

The browser has various UI elements like a search bar, a table icon, and download icons. Below the code editor, the results section shows a single row:

count(i)
1 87575

At the bottom of the results section, a status message reads: "Started streaming 1 record after 40 ms and completed after 41 ms."

Figure 2: Task 1 Output: Total number of intersections.



A screenshot of the Neo4j browser interface. At the top, there is a code editor window containing the following Cypher query:

```
1 MATCH ()-[r:ROAD]->()
2 RETURN count(r) AS total_roads;
```

The browser has various UI elements like a search bar, a table icon, and download icons. Below the code editor, the results section shows a single row:

total_roads
1 199654

At the bottom of the results section, a status message reads: "Started streaming 1 record after 2 ms and completed after 3 ms."

Figure 3: Task 1 Output: Total number of roads.

Although Neo4j relationships are directional, the road network is undirected. Duplicate directional edges were consolidated to represent each road as a single undirected relationship.

2. Task 2: Shortest Path Between Two Intersections

The shortest path between selected intersections was computed using Dijkstra's algorithm via the APOC library. The algorithm minimizes total Euclidean distance and efficiently identifies optimal routes in the large-scale road network.

Step 1: Store Euclidean Distance on ROAD

Euclidean distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
1 MATCH (a:Intersection)-[r:ROAD]->(b:Intersection)
2 WITH a,b,r,
3     sqrt( (toFloat(b.x)-toFloat(a.x))^2 + (toFloat(b.y)-toFloat(a.y))^2 ) AS d
4 SET r.distance = d;
```

Step 2: Create a GDS Graph Projection

```
1 CALL gds.graph.project(
2   'roadGraph',
3   'Intersection',
4   {
5     ROAD: {
6       type: 'ROAD',
7       orientation: 'UNDIRECTED',
8       properties: 'distance'
9     }
10   }
11 );
```

Step 3: Run Dijkstra Shortest Path

The shortest path between selected intersections was computed using Dijkstra's algorithm via the APOC library. The algorithm minimizes total Euclidean distance and efficiently identifies optimal routes in the large-scale road network.

```
1 MATCH (start:Intersection {id: 1}),
2       (end:Intersection {id: 500})
3 CALL apoc.algo.dijkstra(
4   start,
5   end,
```

```

6   'ROAD',
7   'distance'
8 )
9 YIELD path, weight
10 RETURN path, weight;

```

Output Screenshot

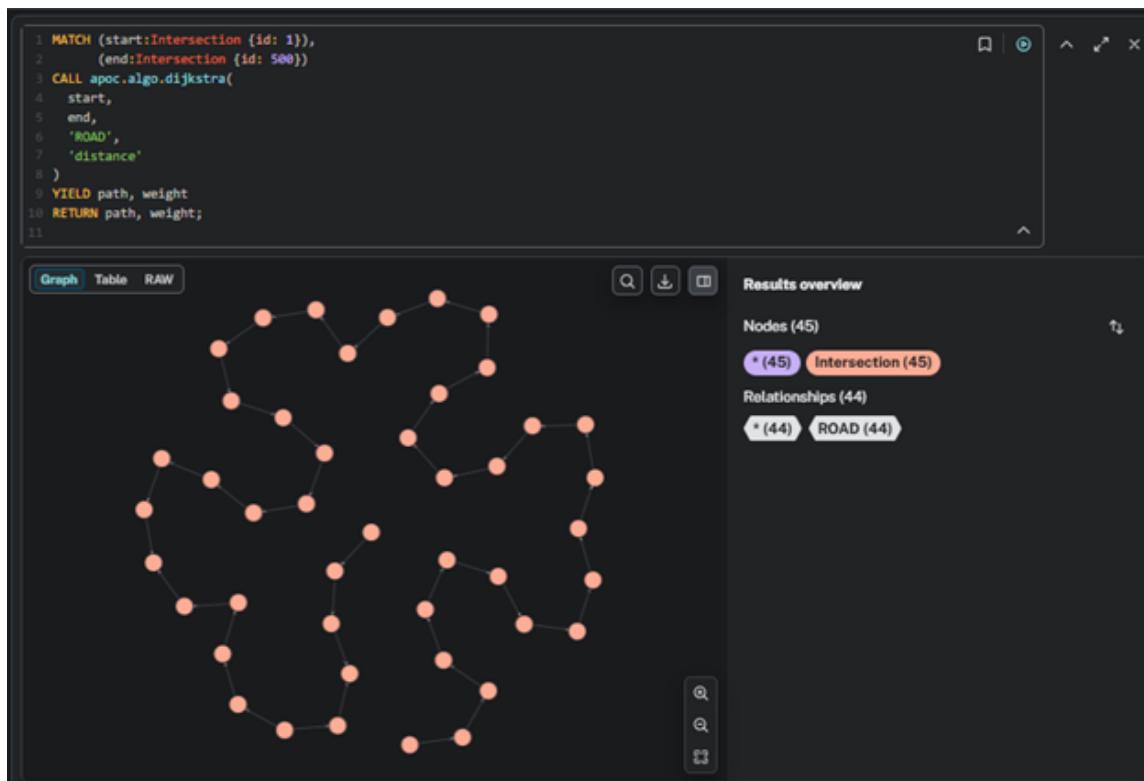


Figure 4: Task 2 Output: Shortest path distance and path nodes Using Dijkstra Algorithm (APOC).

3. Task 3: Intersections with Degree Greater Than a Threshold

Degree analysis was used to identify the most connected intersections. Intersections with high degree represent major road junctions and local connectivity hubs.

Query ($\text{degree} > 4$)

```

1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree

```

```

3 WHERE degree > 4
4 RETURN i.id AS intersection_id, degree
5 ORDER BY degree DESC;

```

Output Screenshot

The screenshot shows a database query results interface. At the top, there is a code editor window containing a Cypher query:

```

1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 WHERE degree > 4
4 RETURN i.id AS intersection, degree
5 ORDER BY degree DESC;
6

```

Below the code editor is a table view. The table has two columns: "intersection" and "degree". The data is as follows:

intersection	degree
2073	6
2581	6
11356	6
84974	6
86470	6
11968	6
53497	6
53658	6
53708	6
53813	6

At the bottom right of the table view, there is a message: "Started streaming 262 records after 139 ms and completed after 302 ms."

Figure 5: Degree Greater Than a Threshold.

Explanation

Degree is the number of roads connected to an intersection. This query identifies highly connected intersections. Common choices are > 3 , > 4 , > 5 . We prefer to pick 4 as 3 given as the example hit the limit which 5000.

Output Screenshot

The screenshot shows a Neo4j browser window with the following details:

- Query:**

```
1 CALL gds.graph.project(
2   'roadGraph',
3   'Intersection',
4   {
5     ROAD: {
6       type: 'ROAD',
7       orientation: 'UNDIRECTED',
8       properties: 'distance'
9     }
10 }
11 );
12 
```
- Table View:** The results are displayed in a table with columns: nodeProjection, relationshipProjection, graphName, nodeCount, relationshipCount, and projectMillis.
- Data:**

nodeProjection	relationshipProjection	graphName	nodeCount	relationshipCount	projectMillis
{ Intersection:{ properties:{}, label:"Intersection" } }	{ ROAD:{ orientation:"UNDIRECTED", aggregation:"DEFAULT", type:"ROAD", properties:{ distance:{ property:"distance", aggregation:"DEFAULT", defaultValue:null } }, indexInverse:false } }	"roadGraph"	87575	242988	1002
- Message:** "Started streaming 1 record after 61 ms and completed after 1,563 ms."

Figure 6: Task 3 Output: Intersections with degree greater than 3.

4. Task 4: Betweenness Centrality Analysis

Betweenness centrality was computed to identify critical intersections in the road network. Intersections with the highest scores occur most frequently on shortest paths between other intersections, indicating their importance as structural bridges in the network. These nodes likely correspond to major road junctions or highway interchanges whose disruption would significantly affect overall connectivity.

Query (Top 10 by Betweenness Centrality)

```
1 CALL gds.betweenness.stream('roadGraph')
2 YIELD nodeId, score
3 RETURN gds.util.asNode(nodeId).id AS intersection_id,
4       score AS betweenness_score
5 ORDER BY betweenness_score DESC
6 LIMIT 10;
```

Output Screenshot

The screenshot shows a database query interface with the following code:

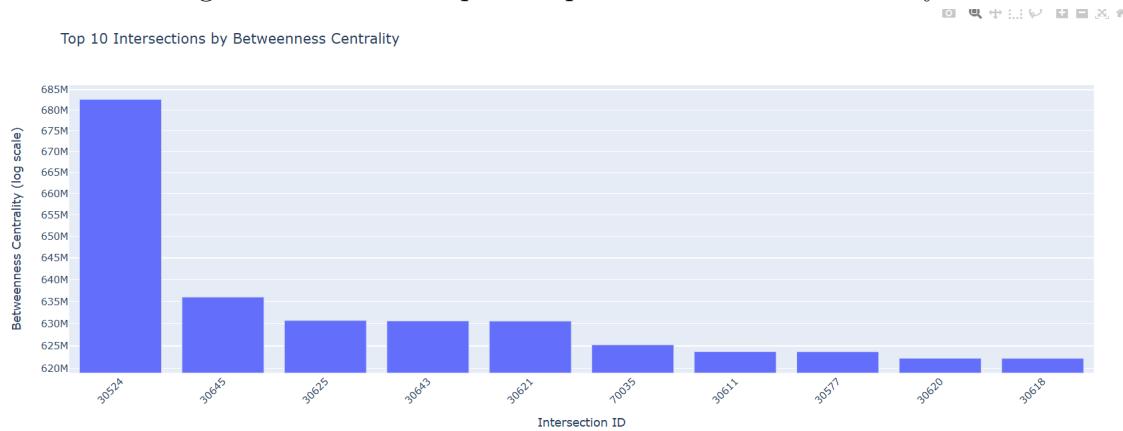
```
1 CALL gds.betweenness.stream('roadGraph')
2 YIELD nodeId, score
3 RETURN gds.util.asNode(nodeId).id AS intersection,
4     score
5 ORDER BY score DESC
6 LIMIT 10;
7
```

The results are displayed in a table:

intersection	score
30524	682642471.506271 2
30645	636018047.070520 4
30625	630714200.1758
30643	630605115.684141 4
30621	630578458.660496 6
70035	625250188.041415 2
30611	623713396.893270 5

At the bottom, it says "Started streaming 10 records after 81 ms and completed after 11 minutes 9 seconds."

Figure 7: Task 4 Output: Top 10 betweenness centrality.



5. Task 5: Dashboard and Key Metrics using Plotly

A Plotly-based dashboard was developed to display key metrics such as total intersections, total roads, and average node degree, providing a high-level overview of the network.

Dashboard Metrics

The dashboard displays:

- Total Intersections
- Total Roads

- Degree Distribution

Output Screenshot(s)



Figure 8: Key metrics

6. Task 6: Degree Distribution

The degree distribution bar chart shows that most intersections have a small number of connected roads, while a small number of intersections act as highly connected hubs. This confirms the sparse nature of the network.

Query

```

1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 RETURN degree, count(i) AS frequency
4 ORDER BY degree;

```

Explanation

This chart shows how connected intersections are. Most intersections usually have low degree, while few may form hubs.

Output Screenshot

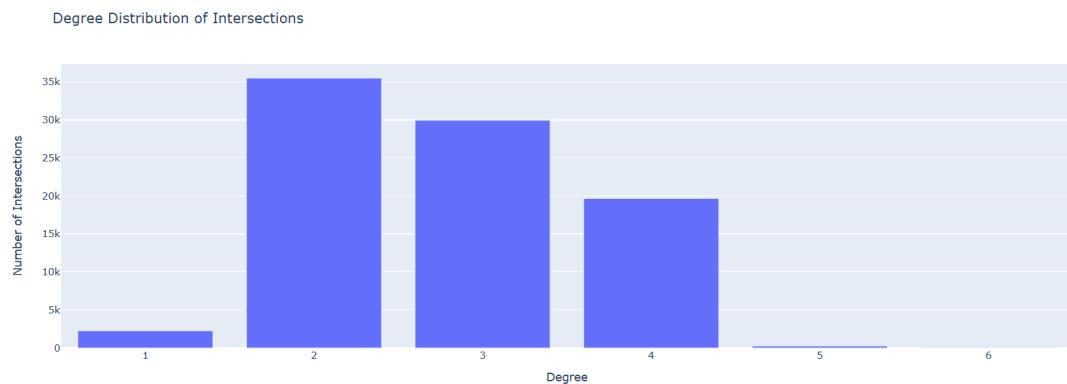


Figure 9: Task 6 Output: Degree distribution of intersections.

7. Task 7: Top 10 Most Connected Intersections

Query

```
1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 RETURN i.id AS intersection_id, degree
4 ORDER BY degree DESC
5 LIMIT 10;
```

Explanation

This identifies the most connected intersections, which may represent important hubs.

Output Screenshot

The screenshot shows a Neo4j browser interface with a query editor at the top containing the following Cypher code:

```

1 MATCH (i:Intersection)-[:ROAD]-()
2 RETURN i.id AS intersection,
3     count(*) AS degree
4 ORDER BY degree DESC
5 LIMIT 10;
6

```

Below the query is a table view with two columns: "intersection" and "degree". The data shows 10 rows, each with an intersection ID and a degree of 6. The IDs listed are 11968, 17714, 6017, 11356, 17391, 2581, 2073, 5831, 9631, and 25259.

At the bottom right of the table area, there is a message: "Started streaming 10 records after 264 ms and completed after 521 ms."

Figure 10: Task 7 Output: Top 10 most connected intersections.

The screenshot shows a Neo4j browser interface with three distinct sections. The top section contains the same Cypher code as Figure 10:

```

1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 WHERE degree > 4
4 RETURN i.id AS intersection, degree
5 ORDER BY degree DESC;
6

```

The middle section displays a table with columns "intersection" and "degree", showing 10 rows where the degree is 6. The intersection IDs are 2073, 2581, 11356, 84974, 86470, 11968, 53497, 53658, 53708, and 53813.

The bottom section contains a single-line Cypher query:

```

1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 WHERE degree > 4
4 RETURN count(i) AS intersections_above_threshold;
5

```

This query returns a single row with the value 262, labeled "intersections_above_threshold".

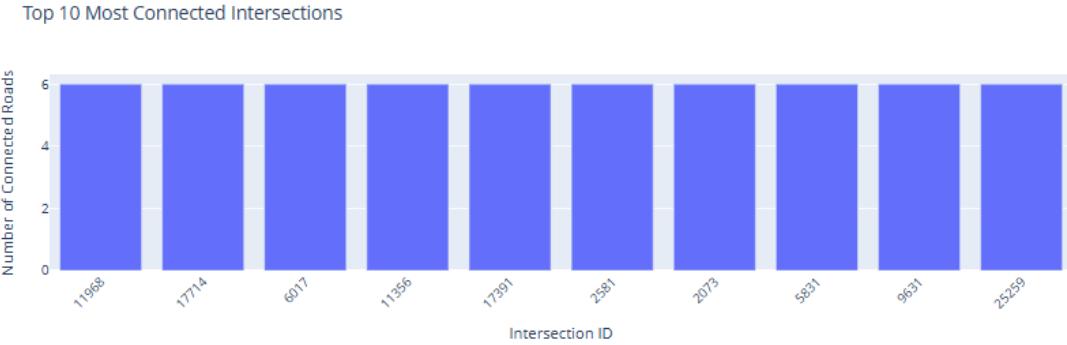


Figure 11: Top 10 most connected nodes.

8. Task 8: Intersection Categories by Degree

Intersections were categorized into low, medium, and high connectivity groups based on node degree. Most intersections fall into the low connectivity category, while highly connected intersections are relatively rare.

Query

```

1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 RETURN
4 CASE
5     WHEN degree <= 2 THEN "Low Connectivity (0-2)"
6     WHEN degree <= 4 THEN "Medium Connectivity (3-4)"
7     ELSE "High Connectivity (5+)"
8 END AS category,
9 count(i) AS number_of_intersections
10 ORDER BY category;
```

Explanation

This groups intersections into low/medium/high connectivity categories, giving a quick summary of connectivity distribution.

Output Screenshot

A screenshot of a database query interface. The top part shows a code editor with the following Cypher query:

```
1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 RETURN degree, count(i) AS num_intersections
4 ORDER BY degree;
5
```

The bottom part shows a table view with the results:

degree	num_intersections
1	2234
2	35487
3	29941
4	19651
5	227
6	35

Started streaming 6 records after 1 ms and completed after 74 ms.

Figure 12: Intersections count.

Output Screenshot

A screenshot of a database query interface. The top part shows a code editor with the following Cypher query:

```
1 Task
2 MATCH (i:Intersection)-[:ROAD]-()
3 WITH i, count(*) AS degree
4 WITH
5   CASE
6     WHEN degree <= 2 THEN 'Low Connectivity'
7     WHEN degree <= 4 THEN 'Medium Connectivity'
8     ELSE 'High Connectivity'
9   END AS category
10 RETURN category, count(*) AS num_intersections
11 ORDER BY num_intersections DESC;
12
```

The bottom part shows a table view with the results:

category	num_intersections
"Medium Connectivity"	49592
"Low Connectivity"	37721
"High Connectivity"	262

Started streaming 3 records after 95 ms and completed after 167 ms.

Figure 13: Intersections categories.

Output Screenshot

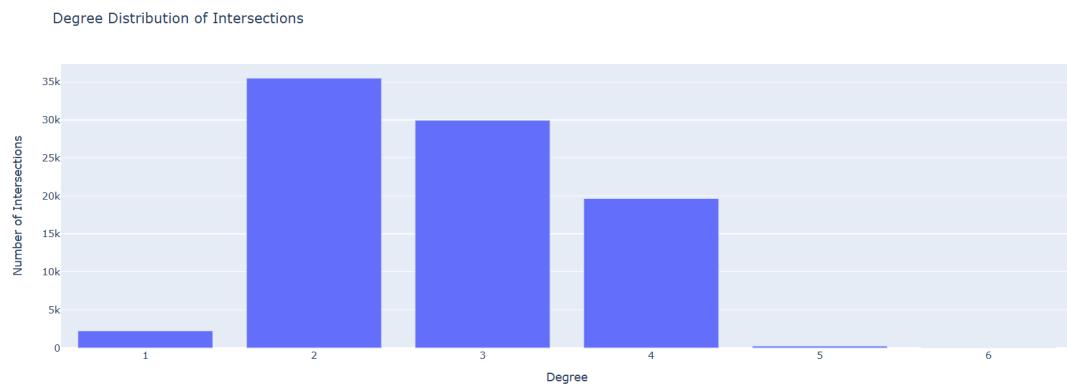


Figure 14: Intersections grouped by degree category.

9. Task 9: Degree Distribution

Query (Same as Task 6)

```
1 MATCH (i:Intersection)-[:ROAD]-()
2 WITH i, count(*) AS degree
3 RETURN degree, count(i) AS frequency
4 ORDER BY degree;
```

Explanation

This is a repeated requirement in the assignment. The output is the same as Task 6.

Output Screenshot

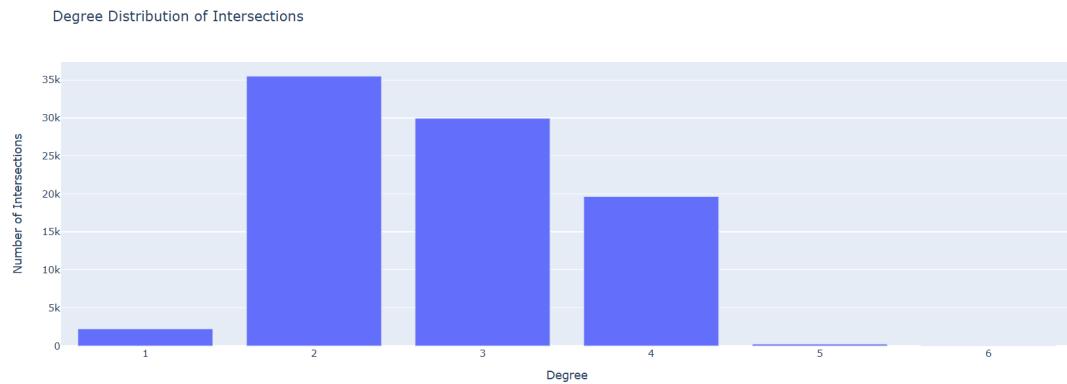


Figure 15: Task 9 Output: Degree distribution (repeated requirement).

Representation using pie chart

The pie chart shows the distribution of intersections by connectivity level. Most intersections fall into the medium connectivity category, indicating that they are connected to a moderate number of roads. A smaller portion of intersections exhibit low connectivity, while only a very small fraction are highly connected hubs. This highlights the sparse structure of the road network, where highly connected intersections are rare but play an important structural role.

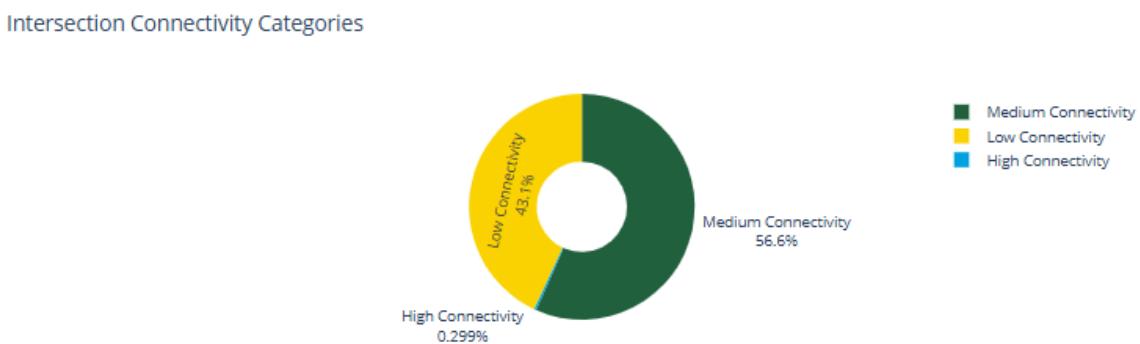


Figure 16: Degree distribution-Pie chart

Conclusion

This project demonstrated the effectiveness of graph databases and graph analytics for analyzing large-scale transportation networks. Using Neo4j, shortest path algorithms, centrality measures, and interactive visualizations, important structural properties of the

U.S. road network were identified. The results highlight the sparse nature of the network and the importance of a small number of highly connected and central intersections.

Neo4j and Graph Data Science (GDS) provide powerful tools for analyzing road networks. Through this assignment, we computed basic graph statistics, shortest paths using weighted edges, degree-based connectivity patterns, and betweenness centrality to identify critical intersections. The Plotly dashboard improved presentation by offering interactive visual summaries of the main network insights.