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

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.

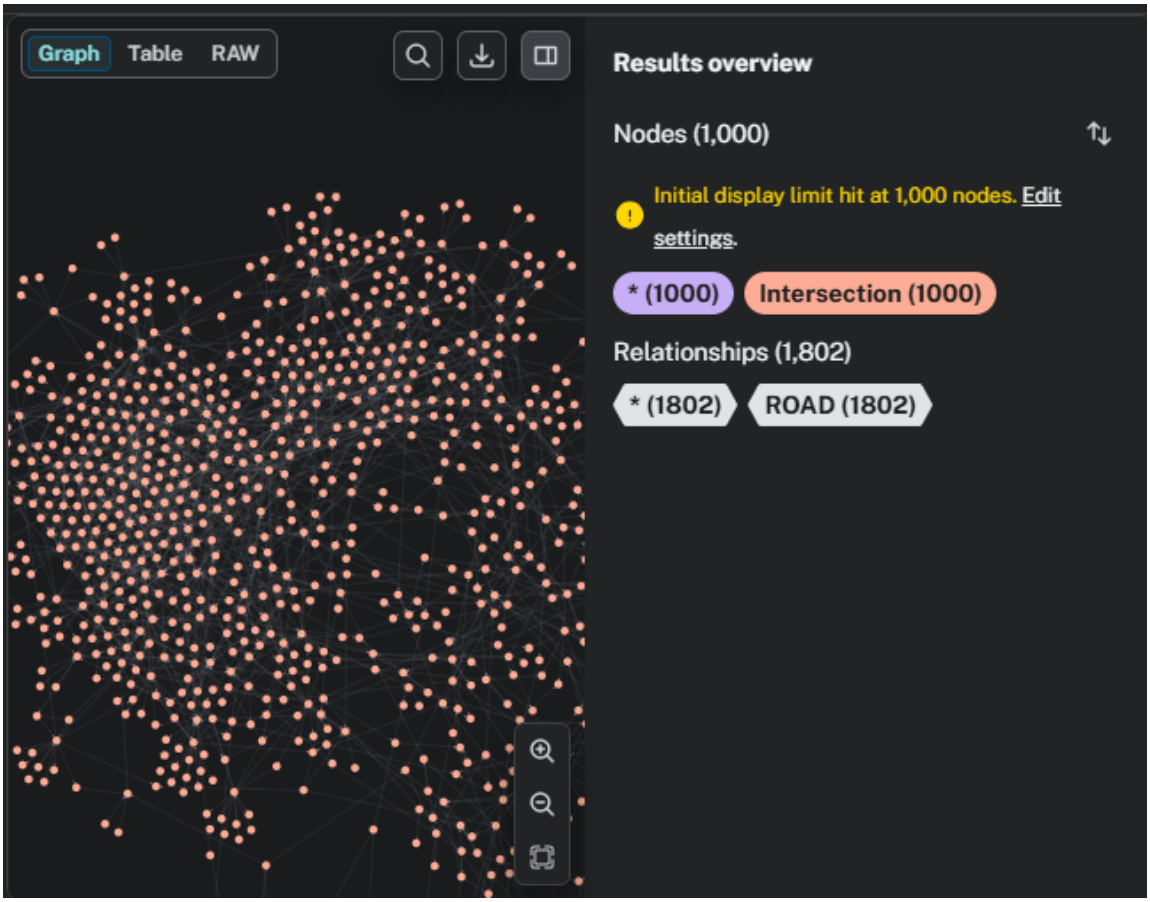


Figure 1: Roads and junctionsnodes

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

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

Dataset Upload Queries

Upload intersections.csv

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

Upload roads.csv

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

1. Task 1: Total Number of Intersections and Roads

Query

```
MATCH (n:Intersection)
RETURN count(n) AS total_intersections;

MATCH ()-[r:ROAD]->()
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

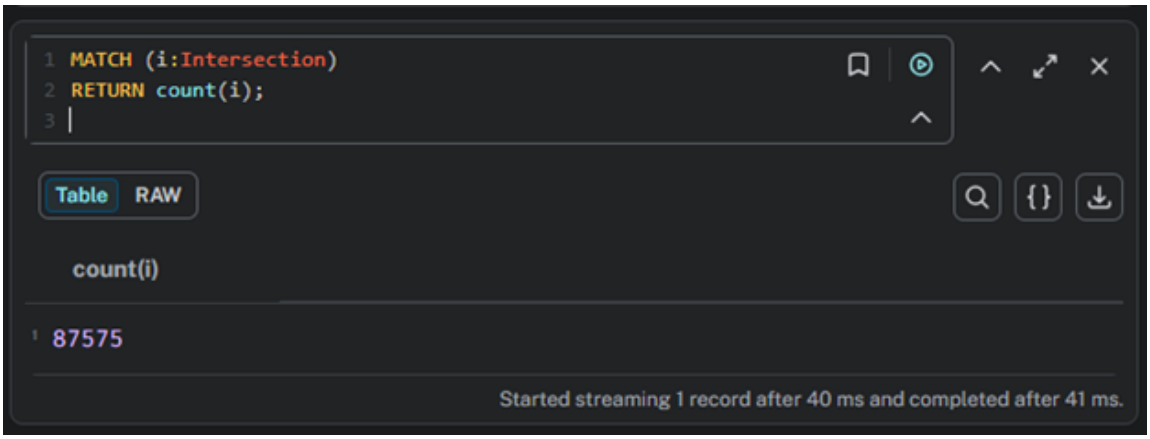


Figure 2: Task 1 Output: Total number of intersections.

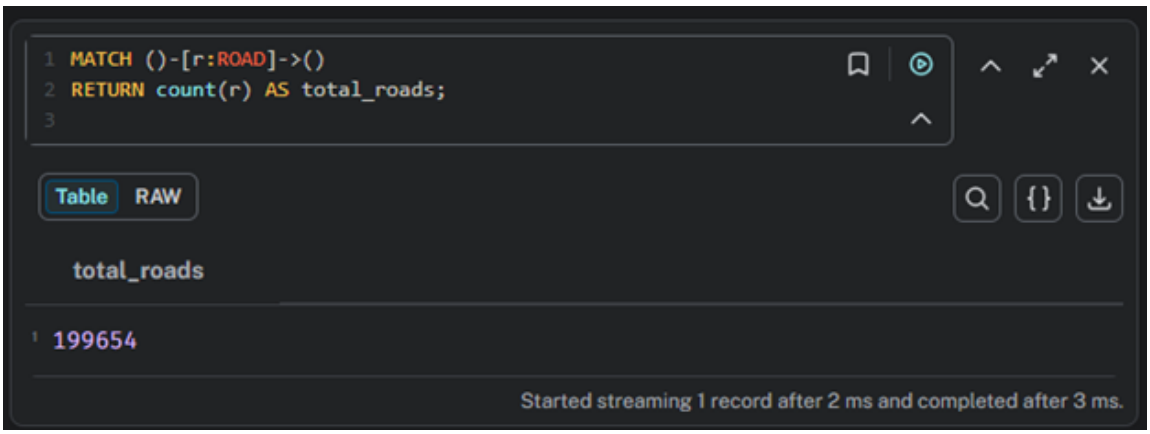


Figure 3: Task 1 Output: Total number of roads.

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:

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

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

Step 2: Create a GDS Graph Projection

```
CALL gds.graph.project(
  'roadGraph',
  'Intersection',
  {
    ROAD: {
      type: 'ROAD',
      orientation: 'UNDIRECTED',
      properties: 'distance'
    }
  }
);
```

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.

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

```
  'ROAD',
  'distance'
)
YIELD path, weight
RETURN path, weight;
```

Output Screenshot

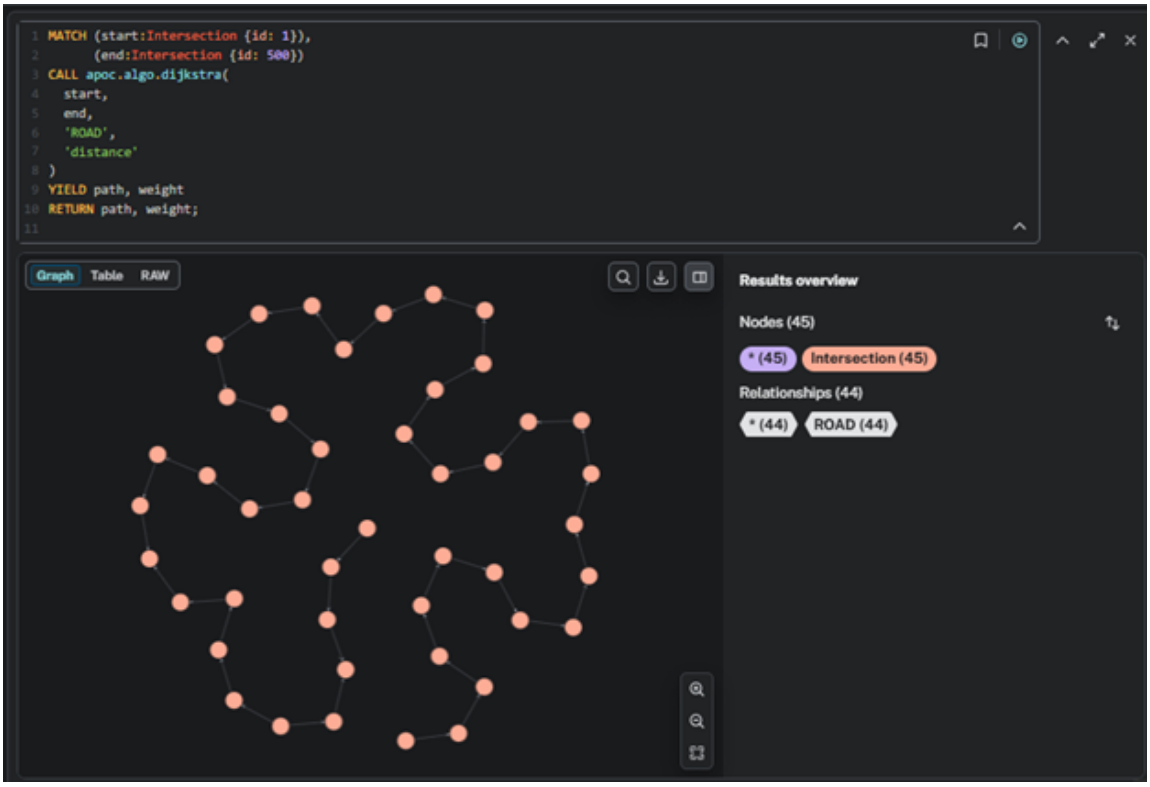


Figure 4: Task 2 Output: Shortest path distance and path nodes Using Dijkstra Algorithm (APOC).

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 (degree > 4)

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

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

Output Screenshot

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

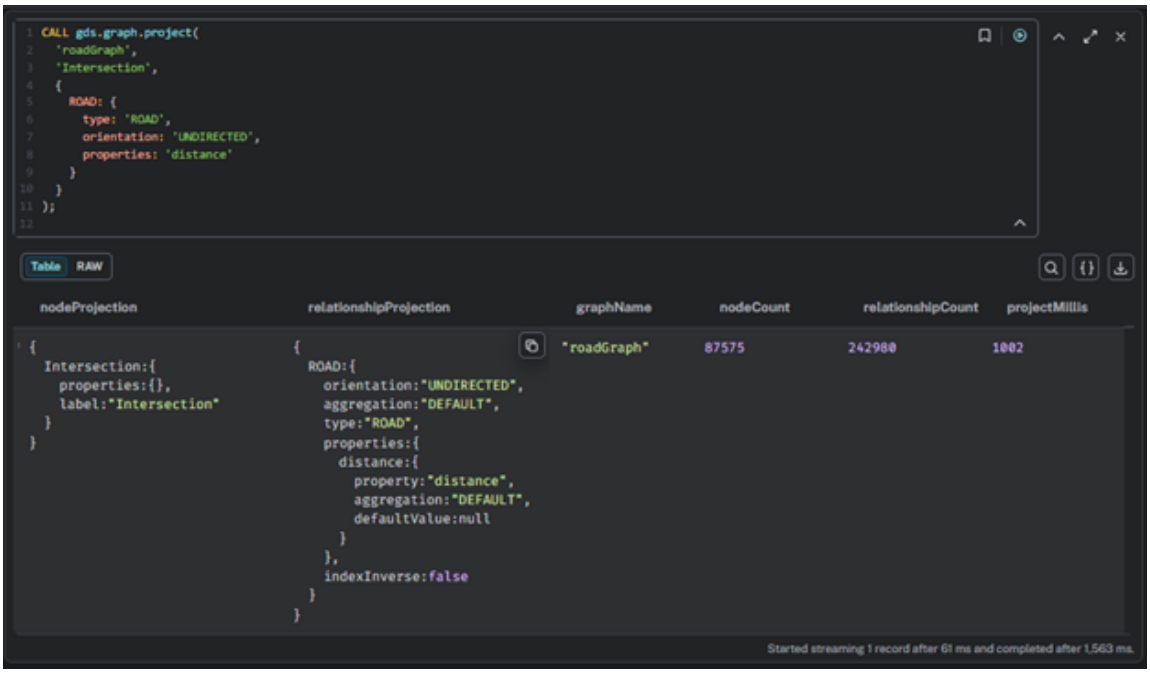


Figure 6: Task 3 Output: Intersections with degree greater than 3.

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)

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

Output Screenshot

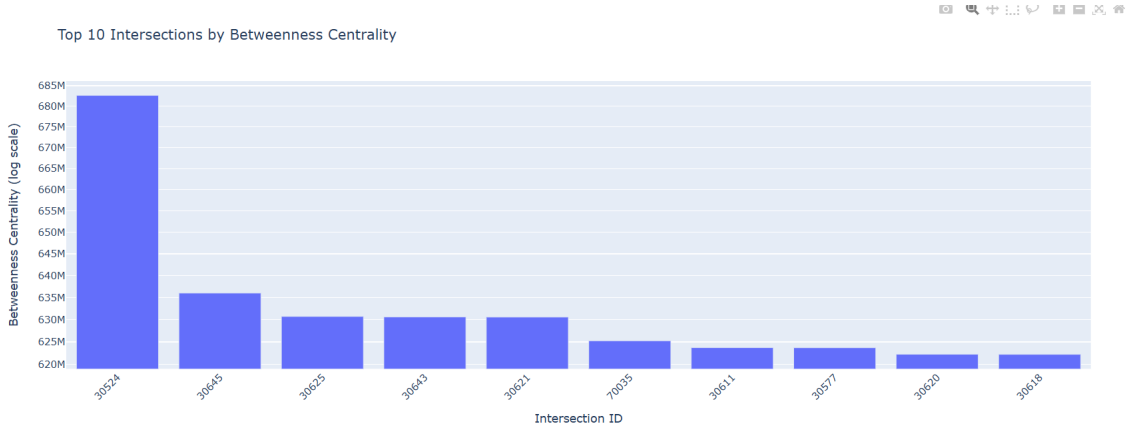
```
CALL gds.betweenness.stream('roadGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).id AS intersection,
       score
ORDER BY score DESC
LIMIT 10;
```

Table RAW

	Intersection	score
1	30524	682642471.5062712
2	30645	636018047.0705204
3	30625	630714200.1758
4	30643	630605115.6841414
5	30621	630578458.6604966
6	70035	625250188.0414152
7	30611	623713396.8932705

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)

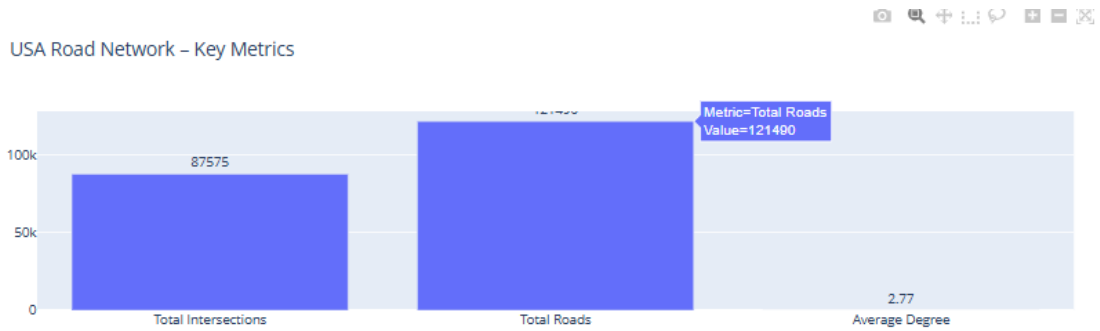


Figure 8: Key metrics

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

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

Explanation

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

Output Screenshot

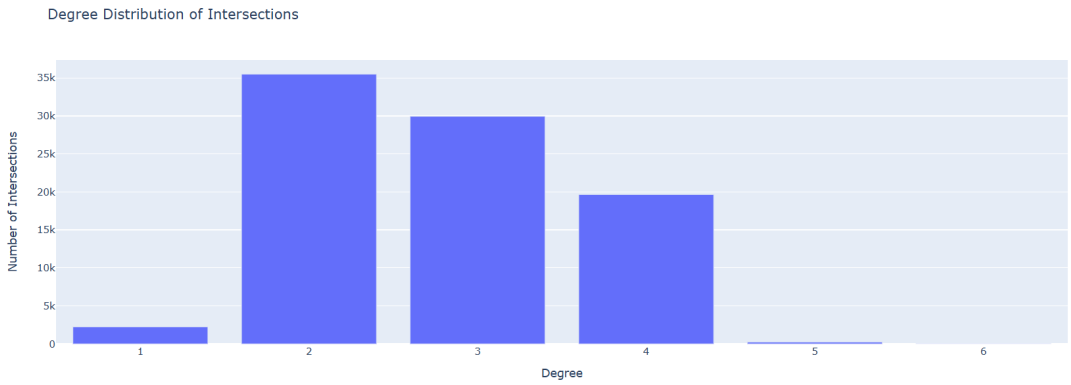


Figure 9: Task 6 Output: Degree distribution of intersections.

7. Task 7: Top 10 Most Connected Intersections

Query

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

Explanation

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

Output Screenshot

```
MATCH (i:Intersection)-[:ROAD]-()
RETURN i.id AS intersection,
       count(*) AS degree
ORDER BY degree DESC
LIMIT 10;
```

Table RAW

	intersection	degree
1	11968	6
2	17714	6
3	6017	6
4	11356	6
5	17391	6
6	2581	6
7	2073	6
8	5831	6
9	9631	6
10	25259	6

Started streaming 10 records after 264 ms and completed after 521 ms.

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

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

Table RAW

	intersection	degree
1	2073	6
2	2581	6
3	11356	6
4	84974	6
5	86470	6
6	11968	6
7	53497	6
8	53658	6
9	53708	6
10	53813	6

Started streaming 262 records after 139 ms and completed after 302 ms.

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

Table RAW

	intersections_abo
1	262

Started streaming 1 record after 90 ms and completed after 207 ms.

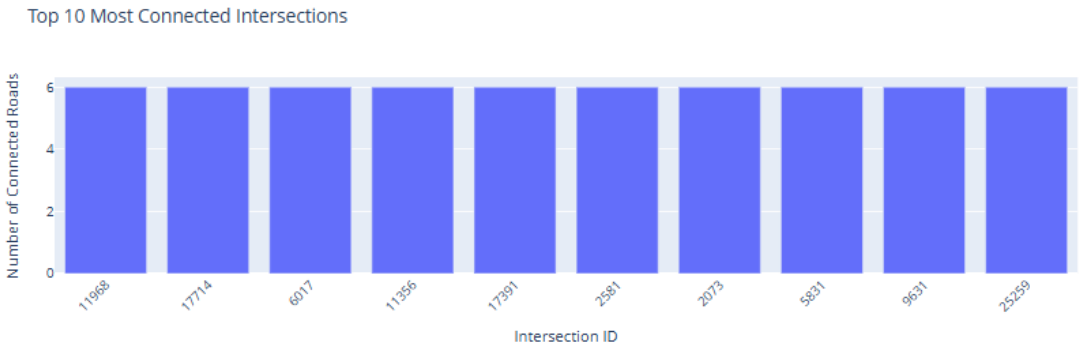


Figure 11: Top 10 most connected nodes.

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

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

Explanation

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

Output Screenshot

Figure 12: Intersections count.

Output Screenshot

Figure 13: Intersections categories.

Output Screenshot

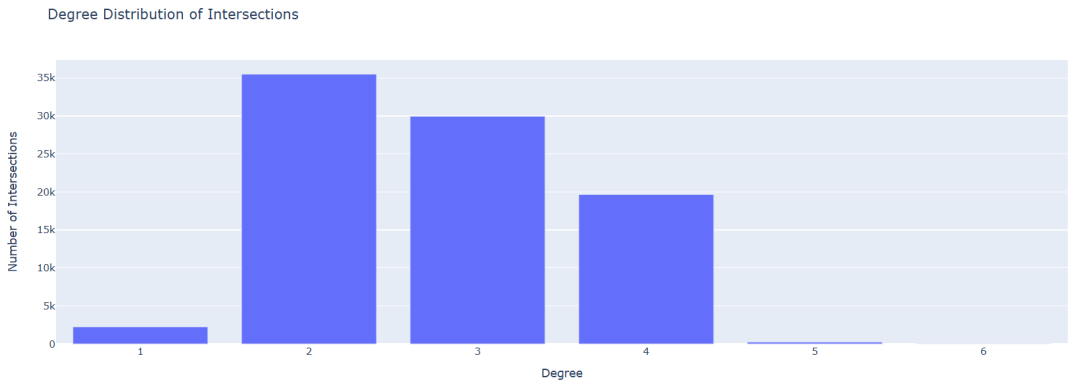


Figure 14: Intersections grouped by degree category.

9. Task 9: Degree Distribution

Query (Same as Task 6)

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

Explanation

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

Output Screenshot

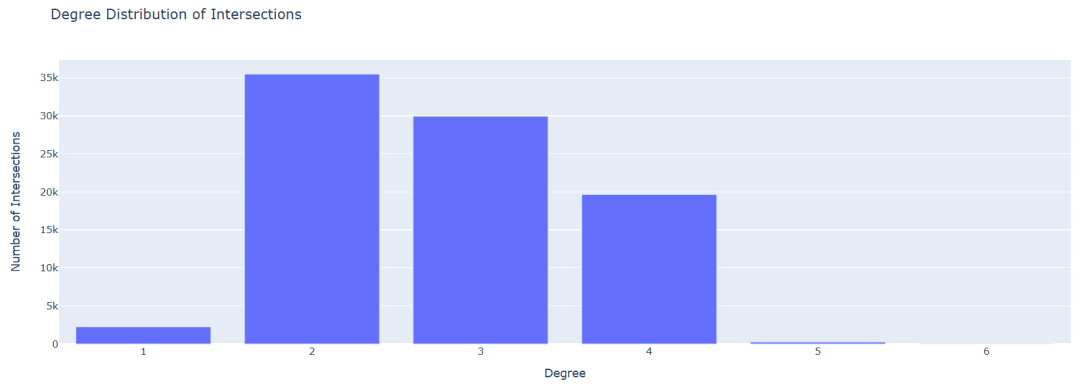


Figure 15: Task 9 Output: Degree distribution (repeated requirement).

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.

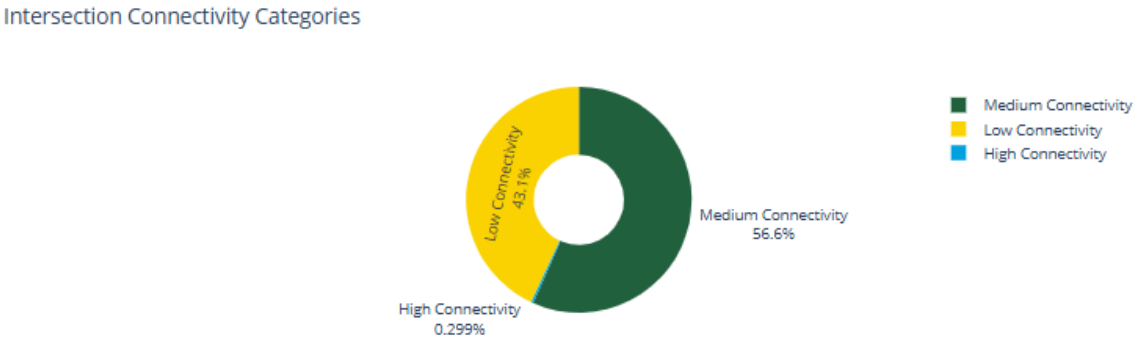


Figure 16: Degree distribution-Pie chart

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.