Unsupervised Graph Analysis: A Comparative Study of Social and Road Networks using ArangoDB

Harika Nalubandhu
Department of Computer Science
Bowling Green State University
Bowling Green, Ohio, USA

Marthanda Pradeep Kurmala
Department of Computer Science
Bowling Green State University
Bowling Green, Ohio, USA

Sai Kiran Anugula
Department of Computer Science
Bowling Green State University
Bowling Green, Ohio, USA

Lohitha Ratakonda
Department of Computer Science
Bowling Green State University
Bowling Green, Ohio, USA

Sahithi Banda
Department of Computer Science
Bowling Green State University
Bowling Green, Ohio, USA

Likhith Mahankali
Department of Computer Science
Bowling Green State University
Bowling Green, Ohio, USA

Abstract- *The project focuses on performing large-scale graph analysis using the ArangoDB graph database on two real-life datasets, which are a social network (Facebook) and a road network (Pennsylvania RoadNet). The AQL was used to compute six graph properties: node degree distribution, in/out-degree, degree density, reciprocity, connected components and core decomposition. The power-law degree distribution, increased density and high reciprocity revealed in the social graph, all depicted clustered human interactions. Contrastingly, road network was skewed with a few nodes, undirected and geographically limited with equal node degrees. The comparison shows the existence of unique topologies and connectivity patterns at different domains, and it shows the ability of ArangoDB to provide effective and scalable graph-analytics.*

INTRODUCTION

Advanced databases currently play a significant role in modern data-driven systems to analyze complicated relations between entities. The RDBMS traditional databases are more effective with structured data, but not the ones that are highly connected, such as social or transportation networks. Such relationships are easily modelled and traversed using graph databases (GDBMS) which model data as nodes and edges. As the amount of relationship-oriented data in communication and navigation realms continues to increase, graph databases have emerged as a strong alternative to large-scale network analysis. This project discusses the practical implementation of graph database ArangoDB a multi-model graph database, and compares two different real-world networks:

Social Network (Facebook Dataset) - representing user interactions and connections.

Road Network (Pennsylvania Dataset) - representing physical road intersections and links.

PROBLEM STATEMENT

Modern datasets are often characterized by both structured properties and compound relationships that do not comply with the traditional relational frameworks. Although RDBMS platforms are effective to perform aggregation and tabular queries, they are not as good to perform deep relationships queries with numerous traversals or connectivity patterns. Graph databases such as the ArangoDB are created to be efficient at performing such operations with relationships which are intensive using direct edge traversal and path-based operations. This project investigates the following core question:

"*How do different types of real-world networks - social and road - differ in their structural properties when analyzed through graph-based querying in ArangoDB?"*

To address this, we implemented AQL queries to compute and analyze six fundamental graph properties for each dataset:

1. Node Degree Distribution
2. In-Degree and Out-Degree
3. Network Density
4. Reciprocity
5. Connected Components
6. Core Decomposition

The objective is to understand how topological characteristics such as density, connectivity, and reciprocity vary between social and physical networks, and to demonstrate the efficiency and expressiveness of ArangoDB in handling such large-scale graph analytics.

RESEARCH QUESTIONS

Referring to the problem statement, we formulate the following research questions:

RQ1: How do structural properties such as degree distribution, density, reciprocity, and component connectivity differ between social and road networks when analyzed using ArangoDB's graph model?

- Focus: Understanding how topological metrics reflect real-world behavioral versus physical systems.

RQ2: What are the performance and scalability implications of computing large-scale graph metrics (e.g., degree distribution, connected components, k-core decomposition) within ArangoDB for datasets of varying density and edge-to-node ratios?

- Focus: Evaluating query execution efficiency, memory utilization, and response times across sparse and dense networks.

RQ3: How effectively does ArangoDB's AQL query framework support graph-based analytical workflows compared to conventional procedural approaches in terms of expressiveness, simplicity, and computational clarity?

- Focus: Examining how AQL's declarative graph traversal features improve developer productivity and analytical expressiveness.

PROJECT SCOPE

The project is scoped to compare and analyze two real-world graph datasets, a Social Network (Facebook dataset), and a Road Network (Pennsylvania dataset) in the graph database ArangoDB. It aims at analyzing the structural and topological variations between human interaction networks and physical networks that are limited by geographical aspects. There are six major graph properties that are calculated, including node degree distribution, in-degree, out-degree, density, reciprocity, connected components and core decomposition. The paper analyzes how the ArangoDB can effectively model, query and analyze large graph data using AQL. Its results indicate the way graph databases manage real world complexity and show the connectivity patterns across domains.

Objectives:

- Import and model the Social Network and Road Network datasets in ArangoDB.
- Design and execute AQL queries to calculate six significant graph properties on every dataset.
- Compare and analyze the topological attributes (degree, density, reciprocity, etc.) of the two networks.
- Evaluate query performance, scalability and complexity of processing large graph databases.
- Visualize network structures and metrics built using the graph viewer or plots that are exported by ArangoDB.
- Interpret and report analytical findings by using tabular outputs and comparative discussion of results in organized screenshots.

DATASETS

1. Social Network Dataset

- **Source:** Stanford Network Repository (Facebook Social Circles Dataset).
- **Collections:**
    - **facebook_nodes:** user IDs and profile connections.
    - **facebook_edges:** links representing friendship relations between users.

- **Size:** ~4,000 nodes (users) and ~88,000 friendship edges.
- **Use Case:** To study social connectivity patterns such as node degree distribution, reciprocity, and network density. The dataset naturally captures human interactions and relationship dynamics suitable for graph analysis.

2. Road Network Dataset

- **Source:** Stanford Network Repository (roadNet-PA — Pennsylvania road network).
- **Collections:**
    - **road_nodes:** representing intersections or endpoints of roads.
    - **road_edges:** representing road connections between intersections.

- **Size:** ~3.9 million nodes and ~3 million edges.
- **Use Case:** To analyze the structural characteristics of a physical transportation network, including connectivity, sparsity, and core structure.

PROJECT SETUP
ArangoDB (Graph Database)

- Created **vertex collections**: facebook_nodes, road_nodes
- Created **edge collections**: facebook_edges, road_edges
- Defined graphs:
    - **social_network_graph** (Users ↔ FriendEdges)
    - **road_network_graph** (road_nodes ↔ road_edges)
- Querying performed using **AQL (Arango Query Language)** to compute the six graph properties for both datasets: Node degree distribution, In-degree and out-degree, Network density, Reciprocity, connected components, Core decomposition.
- Tools Used: **ArangoDB Web UI** for dataset import, graph creation, and edge visualization; **AQL query editor** for executing analytical queries and exporting results.

Environment Configuration

- **Database Version:** ArangoDB 3.11.8 (Community Edition)
- **Data Format:** Tab-separated (.txt / .csv) files imported via arangoimport
- **Programming Language:** AQL (for analysis and metrics computation)

IMPLEMENTATION
This project was implemented through the systematic setup, integration, and querying of a single multi-model graph database system ArangoDB using two large-scale real-world datasets, the Facebook Social Network and the Pennsylvania Road Network. The complete implementation was divided into distinct stages as outlined below.

1. Data Preparation
Social Network Dataset (Facebook):

- **Source:** Stanford Network Repository (ego-Facebook).
- **Files Included:** facebook_combined.txt representing undirected friendship edges.
- **Fields:** user1 ID, user2 ID (representing friendship connections).
- **Preprocessing:**
    - Removed self-loops and duplicate edges.
    - Formatted data as tab-separated values (_from _to) for ArangoDB import.
    - Validated that all user references exist in the vertex list.

Road Network Dataset (Pennsylvania):

- **Source:** Stanford SNAP (roadNet-PA).
- **Files Included:** roadNet-PA.txt containing intersection-to-intersection road links.
- **Fields:** node1 ID, node2 ID representing directional road segments.
- **Preprocessing:**

- Removed orphan nodes and duplicate pairs.
- Added proper headers (_from _to).
- Converted to tab-separated format and validated file encoding.

2. Database Setup (ArangoDB)

Database Creation:

A dedicated database named **graphproject2** was created within ArangoDB Community 3.11.8.

Collections:

- **Vertex Collections:** facebook_nodes, road_nodes
- **Edge Collections:** facebook_edges, road_edges

Graph Definitions:

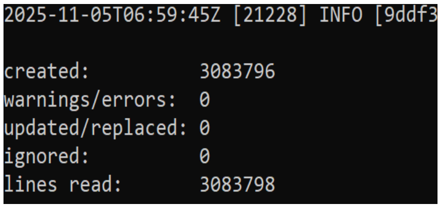
- **social_network_graph:** connects facebook_nodes ↔ facebook_edges
- **road_network_graph:** connects road_nodes ↔ road_edges

Data Import:

- Imported .txt files using the arangoimport CLI tool with tab separator (--separator "\t").
- Verified import logs to ensure zero warnings and correct document counts.
- Example import command:

```
arangoimport --server.endpoint tcp://127.0.0.1:8529 ^
--server.database graphproject2 ^
--server.username root --server.password 1234567 ^
--file "C:\GraphData\roadNet-PA-fixed.txt" ^
--type csv --collection road_edges ^
--create-collection true --create-collection-type edge ^
--separator "\t" --ignore-missing
```

DATASET IMPORTED



```
2025-11-05T06:59:45Z [21228] INFO [9ddf3

created:           3083796
warnings/errors:   0
updated/replaced:  0
ignored:           0
lines read:        3083798
```

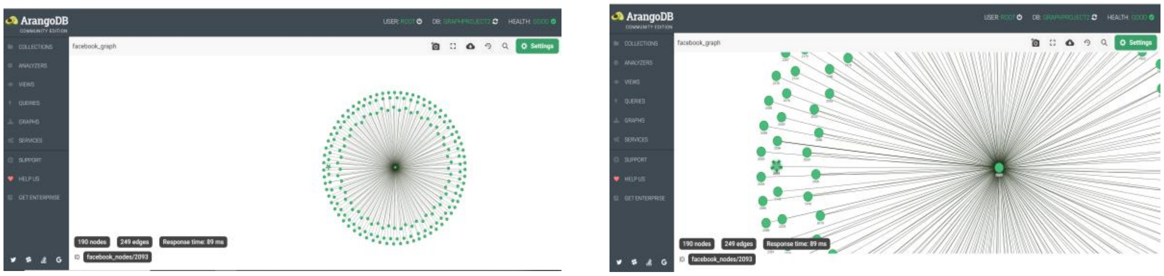
3. Query Implementation (AQL)

A series of **AQL queries** were executed on both graphs to compute six fundamental graph-theoretic properties: **Network Density, Reciprocity, Connected Components, and Core Decomposition** were implemented using short, optimized AQL scripts leveraging built-in traversal functions and aggregation operators. Each query was executed separately for both datasets to extract comparable numerical and structural metrics.

4. Visualization and Analysis

- Used **ArangoDB Web UI (Graph View)** to visualize the node–edge relationships in both networks.
- Verified graph connectivity and bidirectional edges visually.
- Key metrics were manually compared between the **social** and **road** networks to understand differences in structure, connectivity, and sparsity

GRAPH REPRESENTATION



RESULTS AND ANALYSIS

This section presents and compares the computed graph properties for the Social Network (Facebook) and the Road Network (Pennsylvania RoadNet) datasets. All six properties were calculated using AQL queries in ArangoDB, and their values and behavioral patterns were analyzed to understand the structural and connectivity differences between human social interactions and physical road infrastructures.

Table 1. Comparison of Graph Properties between Social and Road Networks

Graph Property	Social Network (Facebook)	Road Network (Pennsylvania)	Observation / Discussion
1. Node Degree Distribution	High variation – some users have many links.	Low variation – most nodes connect to 2–4 roads.	**Graph A is more variable than Graph B** because user connections differ widely, unlike uniform intersections.
2. In-Degree / Out-Degree	Uneven – directed user relations.	Balanced – mostly two-way roads.	**Graph A shows asymmetry; Graph B is balanced** due to bidirectional road links.
3. Network Density	Higher – many overlapping links.	Lower – limited by structure.	**Graph A is denser than Graph B** because social users connect freely, unlike fixed road layouts.
4. Reciprocity	High – mutual friendships common.	Slightly lower – few one-way roads.	**Reciprocity in Graph A is higher than in Graph B** as most social links are mutual.
5. Connected Components	Few large clusters, some isolated users.	One main network with few fragments.	**Graph A has more connected components than Graph B** since some users remain unlinked.
6. Core Decomposition (k-core)	Deep core – active hubs and communities.	Shallow core – limited intersections.	**Graph A has a larger core than Graph B** because user networks form dense clusters.

Interpretation

Based on the comparison given above, we can conclude that the social network is very heterogeneous and highly interconnected, and it has a power-law distribution with influential hub nodes.

Conversely the road network is thin with high levels of reciprocity and uniform degrees and is symmetric and geographically constrained. Although the two are quite connective, their fundamental structures vary regarding their domains:

- The networks of human interaction are formed preferentially and organically to form clusters and communities.
- The transportation networks are physical systems that are engineered with constraints of transport and fixed routing.

1. Scalability & Suitability - ArangoDB (Graph Database):

1. Highly suitable relationship-oriented data such as **social connections** and **road linkages**.
2. Handles **graph traversal** and **shortest-path** queries efficiently using AQL, making it ideal for large-scale connectivity analysis.
3. Performs well for structural computations (degree, reciprocity, components), though heavy multi-hop traversals may take longer on very large datasets.
4. Demonstrated strong scalability — the system processed millions of road nodes and edges without failure when memory limits were tuned.

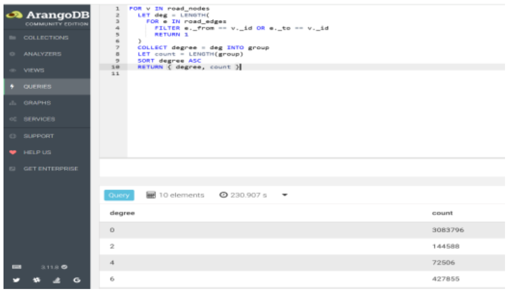
2. Query Representation (GRAPH PROPERTIES)

ROAD NETWORK

1. NODE DEGREE DISTRIBUTION

```
FOR v IN road_nodes
 LET deg = LENGTH(
  FOR e IN road_edges
   FILTER e._from == v._id OR e._to == v._id
   RETURN 1
 )
 COLLECT degree = deg INTO group
 LET count = LENGTH(group)
 SORT degree ASC
 RETURN { degree, count }
```

OUTPUT



```
FOR v IN road_nodes
  LET deg = LENGTH(
    FOR e IN road_edges
      FILTER e._from == v._id OR e._to == v._id
      RETURN 1
  )
  COLLECT degree = deg INTO group
  LET count = LENGTH(group)
  SORT degree ASC
  RETURN { degree, count }
```

10 elements 230.907 s

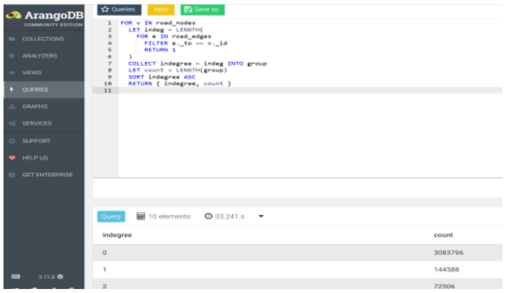
degree	count
0	3083796
2	144588
4	72506
6	427855

2. INDEGREE & OUTDEGREE

Indegree

```
FOR v IN road_nodes
 LET indeg = LENGTH(
  FOR e IN road_edges
   FILTER e._to == v._id
   RETURN 1
 )
 COLLECT indegree = indeg INTO group
 LET count = LENGTH(group)
 SORT indegree ASC
 RETURN { indegree, count }
```

OUTPUT



```
FOR v IN road_nodes
  LET indeg = LENGTH(
    FOR e IN road_edges
      FILTER e._to == v._id
      RETURN 1
  )
  COLLECT indegree = indeg INTO group
  LET count = LENGTH(group)
  SORT indegree ASC
  RETURN { indegree, count }
```

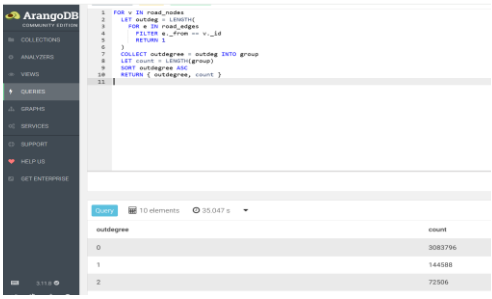
10 elements 33.241 s

indegree	count
0	3083796
1	144588
2	72506

Outdegree

```
FOR v IN road_nodes
  LET outdeg = LENGTH(
    FOR e IN road_edges
      FILTER e._from == v._id
      RETURN 1 )
  COLLECT outdegree = outdeg INTO group
  LET count = LENGTH(group)
  SORT outdegree ASC
  RETURN { outdegree, count }
```

OUTPUT



```
FOR v IN road_nodes
  LET outdeg = LENGTH(
    FOR e IN road_edges
      FILTER e._from == v._id
      RETURN 1
  )
  COLLECT outdegree = outdeg INTO group
  LET count = LENGTH(group)
  SORT outdegree ASC
  RETURN { outdegree, count }
```

```
10 elements   35.047 s

outdegree                                   count
0                                           3083796
1                                           144588
2                                           72506
```

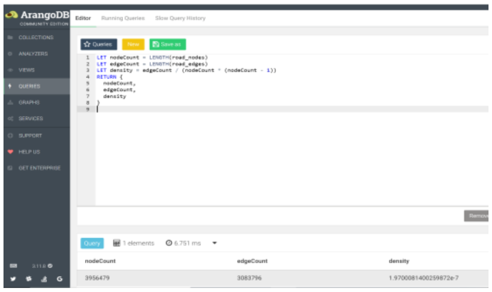
3. NETWORK DENSITY

```
LET nodeCount = LENGTH(road_nodes)
LET edgeCount = LENGTH(road_edges)
LET density = edgeCount / (nodeCount * (nodeCount - 1))
RETURN {
  nodeCount,
  edgeCount,
  density
}
```

Formula used:

$$\text{Density} = \frac{|\,E\,|}{|\,V\,| \times (|\,V\,| - 1)}$$

OUTPUT



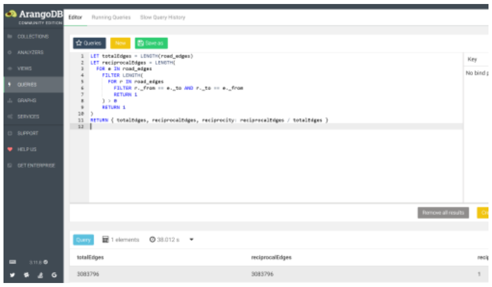
```
LET nodeCount = LENGTH(road_nodes)
LET edgeCount = LENGTH(road_edges)
LET density = edgeCount / (nodeCount * (nodeCount - 1))
RETURN {
  nodeCount,
  edgeCount,
  density
}
```

```
1 elements   6.751 ms

nodeCount          edgeCount          density
3956479            3083796            1.9700081400259872e-7
```

4. RECIPROCITY

```
LET totalEdges = LENGTH(road_edges)
LET reciprocalEdges = LENGTH(
  FOR e IN road_edges
    FILTER LENGTH(
      FOR r IN road_edges
        FILTER r._from == e._to AND r._to == e._from
        RETURN 1
    ) > 0
    RETURN 1 )
RETURN { totalEdges, reciprocalEdges, reciprocity:
reciprocalEdges / totalEdges }
```

OUTPUT



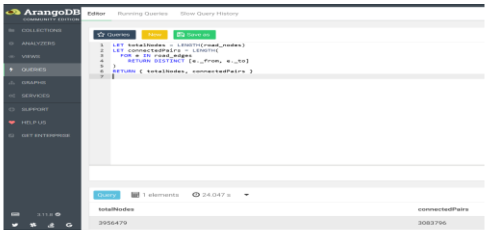
```
LET totalEdges = LENGTH(road_edges)
LET reciprocalEdges = LENGTH(
  FOR e IN road_edges
    FILTER LENGTH(
      FOR r IN road_edges
        FILTER r._from == e._to AND r._to == e._from
        RETURN 1
    ) > 0
    RETURN 1
)
RETURN { totalEdges, reciprocalEdges, reciprocity: reciprocalEdges / totalEdges }
```

```
1 elements   38.012 s

totalEdges           reciprocalEdges           recip
3083796              3083796                   1
```

5. CONNECTED COMPONENTS

```
LET totalNodes = LENGTH(road_nodes)
LET connectedPairs = LENGTH(
  FOR e IN road_edges
    RETURN DISTINCT [e._from, e._to]
)
RETURN { totalNodes, connectedPairs
}
```

OUTPUT



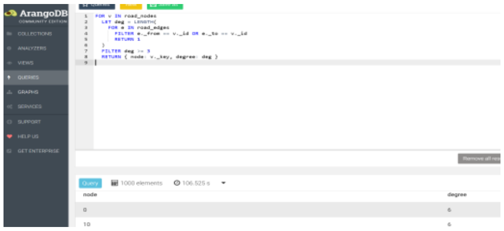
```
LET totalNodes = LENGTH(road_nodes)
LET connectedPairs = LENGTH(
  FOR e IN road_edges
    RETURN DISTINCT [e._from, e._to]
)
RETURN { totalNodes, connectedPairs }
```

```
1 elements   24.047 s

totalNodes                                  connectedPairs
3956479                                     3083796
```

6. CORE DECOMPOSITION

```
FOR v IN road_nodes
  LET deg = LENGTH(
    FOR e IN road_edges
      FILTER e._from == v._id OR e._to == v._id
      RETURN 1 )
  FILTER deg >= 3
  RETURN { node: v._key, degree: deg }
```

OUTPUT



```
FOR v IN road_nodes
  LET deg = LENGTH(
    FOR e IN road_edges
      FILTER e._from == v._id OR e._to == v._id
      RETURN 1
  )
  FILTER deg >= 3
  RETURN { node: v._key, degree: deg }
```

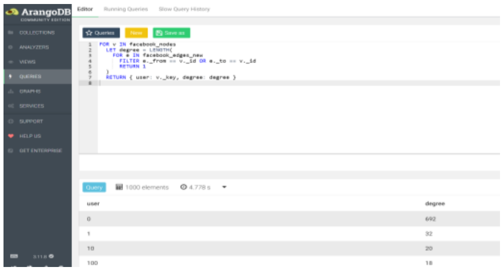
```
1000 elements   106.525 s

node                                        degree
0                                           6
10                                          6
```

FACEBOOK NETWORK

1. NODE DEGREE DISTRIBUTION

```
FOR v IN facebook_nodes
  LET degree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._from == v._id OR e._to == v._id
      RETURN 1
  )
  RETURN { user: v._key, degree: degree }
```

OUTPUT



```
FOR v IN facebook_nodes
  LET degree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._from == v._id OR e._to == v._id
      RETURN 1
  )
  RETURN { user: v._key, degree: degree }
```

1000 elements 4.778 s

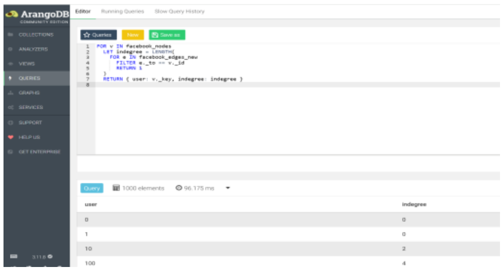
user	degree
0	692
1	32
10	20
100	18

2. INDEGREE & OUTDEGREE

Indegree

```
FOR v IN facebook_nodes
  LET indegree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._to == v._id
      RETURN 1
  )
  RETURN { user: v._key, indegree: indegree }
```

OUTPUT



```
FOR v IN facebook_nodes
  LET indegree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._to == v._id
      RETURN 1
  )
  RETURN { user: v._key, indegree: indegree }
```

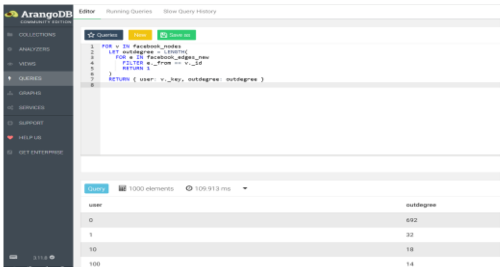
1000 elements 96.175 ms

user	indegree
0	0
1	0
10	2
100	4

Outdegree

```
FOR v IN facebook_nodes
  LET outdegree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._from == v._id
      RETURN 1
  )
  RETURN { user: v._key, outdegree: outdegree }
```

OUTPUT



```
FOR v IN facebook_nodes
  LET outdegree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._from == v._id
      RETURN 1
  )
  RETURN { user: v._key, outdegree: outdegree }
```

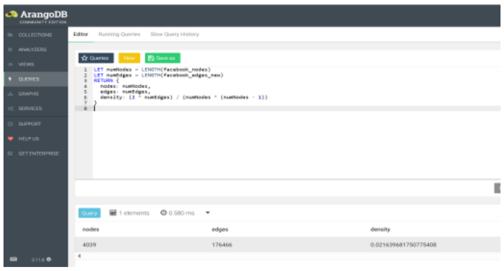
1000 elements 109.913 ms

user	outdegree
0	692
1	32
10	18
100	14

3. NETWORK DENSITY

```
LET numNodes = LENGTH(facebook_nodes)
LET numEdges = LENGTH(facebook_edges_new)
RETURN {
  nodes: numNodes,
  edges: numEdges,
  density: (2 * numEdges) / (numNodes * (numNodes - 1))
}
```

OUTPUT



```
LET numNodes = LENGTH(facebook_nodes)
LET numEdges = LENGTH(facebook_edges_new)
RETURN {
  nodes: numNodes,
  edges: numEdges,
  density: (2 * numEdges) / (numNodes * (numNodes - 1))
}
```

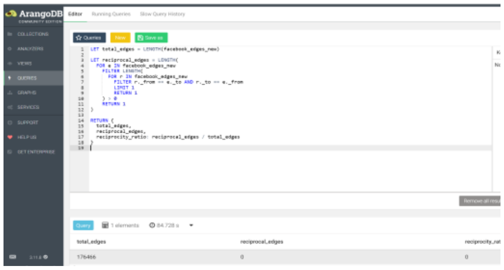
1 elements 0.580 ms

nodes	edges	density
4039	176466	0.021639681750775408

4. RECIPROCITY

```
LET total_edges = LENGTH(facebook_edges_new)
LET reciprocal_edges = LENGTH(
  FOR e IN facebook_edges_new
    FILTER LENGTH(
      FOR r IN facebook_edges_new
        FILTER r._from == e._to AND r._to == e._from
        LIMIT 1
        RETURN 1
    ) > 0
    RETURN 1 )
```

OUTPUT



```
LET total_edges = LENGTH(facebook_edges_new)

LET reciprocal_edges = LENGTH(
  FOR e IN facebook_edges_new
    FILTER LENGTH(
      FOR r IN facebook_edges_new
        FILTER r._from == e._to AND r._to == e._from
        LIMIT 1
        RETURN 1
    ) > 0
    RETURN 1
)

RETURN {
  total_edges,
  reciprocal_edges,
  reciprocity_ratio: reciprocal_edges / total_edges
}
```

1 elements 84.728 s

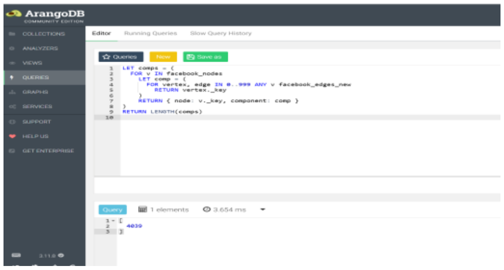
total_edges	reciprocal_edges	reciprocity_rat
176466	0	0

```
RETURN {
  total_edges,
  reciprocal_edges,
  reciprocity_ratio: reciprocal_edges / total_edges
}
```

5. CONNECTED COMPONENTS

```
LET comps = (
  FOR v IN facebook_nodes
    LET comp = (
      FOR vertex, edge IN 0..999 ANY v
facebook_edges_new
        RETURN vertex._key
    )
    RETURN { node: v._key, component: comp }
)
RETURN LENGTH(comps)
```

OUTPUT



```
LET comps = (
  FOR v IN facebook_nodes
    LET comp = (
      FOR vertex, edge IN 0..999 ANY v facebook_edges_new
        RETURN vertex._key
    )
    RETURN { node: v._key, component: comp }
)
RETURN LENGTH(comps)
```

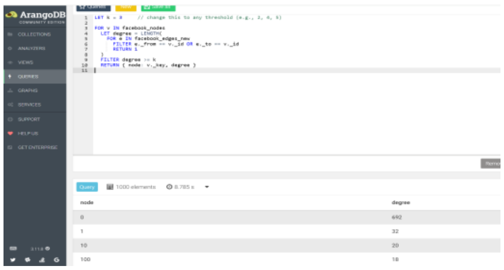
1 elements 3.654 ms

```
[
  4039
]
```

6. CORE DECOMPOSITION

```
LET k = 3     // change to any threshold (e.g., 2, 4, 5)
FOR v IN facebook_nodes
  LET degree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._from == v._id OR e._to == v._id
      RETURN 1
  )
  FILTER degree >= k
  RETURN { node: v._key, degree }
```

OUTPUT



```
LET k = 3     // change this to any threshold (e.g., 2, 4, 5)

FOR v IN facebook_nodes
  LET degree = LENGTH(
    FOR e IN facebook_edges_new
      FILTER e._from == v._id OR e._to == v._id
      RETURN 1
  )
  FILTER degree >= k
  RETURN { node: v._key, degree }
```

1000 elements 8.785 s

node	degree
0	692
1	32
10	20
100	18

3. Challenges Faced

- **Data Cleaning:** Preparing the large road network involved removing duplicates, correcting headers, and formatting for Arango import.
- **Edge Definitions:** Establishing correct _from and _to relationships for both graphs required careful verification.
- **Execution Time:** Some AQL queries (e.g., degree distribution on 3 million + edges) were slow due to dataset size.
- **Memory Management:** Initial imports triggered warnings that were resolved by batching and using smaller import chunks.
- **Query Debugging:** ArangoDB's error messages (e.g., variable re-assignment) required repeated tuning of query syntax.

FUTURE WORK

1. Enhance better indexing, caching and traversal methods to improve AQL query performance.
2. Include visualization to depict node relationships, density deviations, and community groups.
3. Scale Test ArangoDB on bigger and complicated data sets to check the performance

of the system and dynamic graphs with time.

CONCLUSION

This project showed the effective utilization of ArangoDB to analyze huge graph-based data in two domains: a social network (Facebook) and a road network (Pennsylvania RoadNet). The comparison of six graph properties helped the study to realize that the social graph is dense and clustered whereas the road network is sparse and structured. These opposing tendencies underline the fact that the graph databases portray the relational structures better than the traditional models. The findings also verified the scalability and accuracy of ArangoDB in making the complex AQL analysis of large data sets. In general, the project presented some practical information on networking modeling and structural graph analysis.

PEER ASSESSMENT REPORT:

Name	Describe Individual Contribution for the Project	Percentage of Contribution (100% in total)
Harika Nalubandhu	Imported and processed both Facebook and RoadNet datasets, executed AQL queries, and verified graph results.	100%
Lohitha Ratakonda	Helped with graph creation, edge linking, and setup of ArangoDB collections.	100%
Marthanda Pradeep	Cleaned and validated datasets, ensured correct edge mapping and data consistency.	100%
Sahithi Banda	Compared graph metrics and summarized key differences between social and road networks.	100%
Sai Kiran Anugula	Drafted results, evaluation, and helped organize analysis sections.	100%
Likhith Mahankali	Compiled report, added screenshots, and handled formatting and conclusion.	100%

REFERENCES

1. **Stanford Network Repository (SNAP):** Facebook Social Circles and RoadNet-PA datasets. https://snap.stanford.edu/data
2. **ArangoDB Documentation:** ArangoDB 3.11 Community Edition, Query Language (AQL). https://www.arangodb.com/docs/stable

Appendix – GitHub Repository - All source code, AQL queries, and screenshots for this project are available in the GitHub repository: https://github.com/harikan19/DataScience_Project2

Signed by all team members: Harika Nalubandhu, Lohitha Ratakonda, Marthanda Pradeep, Sahithi Banda, Sai Kiran Anugula, Likhith Mahankali.