### **Metrics Evaluated for Various Graph Databases**

**1. Feature / Characteristics Comparison**

* Some of the literature we analysed did not go for a performance-based evaluation, rather the features and characteristics that each graph database had. These include the following features:
  + Data Storing Features:
    - Main memory
    - External memory
    - Backend storage
    - Indexes
  + Operation and Manipulation Features:
    - Data definition language (allows modification of the schema by adding, updating, or deleting objects)
    - Data manipulation language (allows to insert, delete and update data in the database)
    - Query language (data retrieval by query expression)
    - Application Program Interfaces (APIs)
    - Graphical User Interfaces (GUI)
  + Support for Graph Data Structures:
    - Simple Graphs
    - Hypergraphs
    - Nested graphs
    - Attributed Graphs
    - Node labelled
    - Node attribution
    - Directed edges
    - Edge labelled
    - Edge attributions
  + Flexible Schema
  + Sharding (breaking up a large data set across several shards)
  + Backups
  + Scalability: vertical (adding physical resources), or horizontal (adding more instances / nodes)
  + Cloud Ready
  + Multi-model and multi-architecture
  + Language, Platform, Cost, Import and Export styles / availability / formats

**2. Performance Based Evaluation:**

* These types of literature conducted a performance-based evaluation based on some metrics and features. They are listed as below where mostly each feature or metric was evaluated based on its execution time:
  + Installation and setup of the graph data base time
  + Data set loading time / Initial graph construction time
    - Importing data set files
    - Configuring the schema
    - Optimizing database settings for performance
  + Update rate in edges per second
  + Query Execution Time
    - Complex read-only queries
    - Simple read-only queries
    - Transactional updates (inserting new data)
  + Memory usage at difference scale factors of the data
    - CPU usage
    - RAM usage
  + Index based queries:
    - Creating unique indexes for each node and performing query with and without an index (seemed redundant though and was not able to understand this fully)
  + Single Source Shortest Path (SSSP) (BFS and Dijkstra)
  + Shiloach Vishkin (SV) connected components algorithm
  + Connected Components using in-and-out-going edges
  + Page Rank (PR) in the vertex-parallel Bulk Synchronous Parallel power-iteration style (dk what this is, will have to go over it)
  + Breadth-First Search up to k-depth
  + Set of edge insertions and deletions in parallel
  + Build subgraphs in the neighbourhood of a node, computing k-hops operations
  + Traversal over the whole graph. Traversed edges per second estimated using graph operation to calculate betweenness centrality

**3. New Features / Metrics that we can consider:**

* Most of the above metrics that have been listed are a culmination of metrics used in about 10 papers. Papers that focused on comparison of features or characteristics did not go for performance evaluation. However, some papers that did performance evaluation also compared some features.
* Including the above features / characteristics and performance metrics, we can also include the following which were not seen in the literature:
  + OS that they are available for
  + On-going support
  + Worldwide users / community support
  + Integration with other tools such as data science tools, ETL pipelines, web frameworks or support with other languages / drivers etc
  + Ability to make apps with
  + Graph Algorithms:
    - Shortest Path using Dijkstra, A\*, Bellman Ford, Floyd Warshall
    - Minimum Spanning Tree
    - Network Flow algorithms (Ford Fulkerson)
    - Connectivity Algorithms
    - Centrality Measures (degree centrality, closeness centrality, betweenness centrality, eigenvector centrality)
    - Community Detection Algorithms (Louvain method, label propagation, weakly connected components, triangle count)
    - Clustering Coefficient
  + Node embeddings
  + Machine Learning Capabilities
    - Link Prediction
    - Node Classification
    - Graph Embeddings
  + Knowledge Graphs support
  + LLMs (Large language models) integration / support