**# Data Exploration / Insightful queries**

**# Query 1**

MATCH (b:Book) <-[:REVIEWS] – (r:User)

RETURN b.book\_id, count(r.rating) AS total\_ratings

ORDER BY total\_ratings desc

LIMIT 10

**# Query 2**

MATCH (p1:User)-[:REVIEWS]->(b:Book)<-[:WRITES]-(a:Author)

RETURN p1.user\_id AS user, count(distinct a.author) AS authors\_read

ORDER BY authors\_read DESC

LIMIT 10

**# Query 3**

MATCH (p1:User)-[:REVIEWS]->(b:Book)<-[:REVIEWS]-(p2:User)

WHERE id(p1) < id(p2)

RETURN p1.user\_id AS user, count(p2) AS connected\_users

ORDER BY connected\_users DESC

LIMIT 10

**# Projection 1**

CALL gds.graph.create.cypher(

'user-to-user-relationship',

'MATCH (p:User) RETURN id(p) AS id',

'MATCH (p1:User)-[:REVIEWS]->(b:Book)<-[:REVIEWS]-(p2:User) WHERE id(p1) < id(p2) RETURN id(p1) AS source, id(p2) AS target')

**# Graph Algorithms**

**# PageRank -**

CALL gds.pageRank.stream('user-to-user-relationship')

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).user\_id AS user, score

ORDER BY score DESC, user ASC

**# Closeness Centrality**

CALL gds.alpha.closeness.stream('user-to-user-relationship')

YIELD nodeId, centrality

RETURN gds.util.asNode(nodeId).user\_id AS user, centrality

ORDER BY centrality DESC, user ASC

**# Degree Centrality**

CALL gds.degree.stream('user-to-user-relationship')

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).user\_id AS user, score

ORDER BY score DESC, user ASC

**# Louvain Community Detection Algorithm**

CALL gds.louvain.stream('user-to-user-relationship')

YIELD nodeId, communityId, intermediateCommunityIds

RETURN gds.util.asNode(nodeId).user\_id AS user, communityId, intermediateCommunityIds

ORDER BY user ASC

**# Projection 2**

CALL gds.graph.create.cypher(

'user-to-author-relationship',

'MATCH (p:User) RETURN id(p) AS id',

'MATCH (p1:User)-[:REVIEWS]->(b1:Book)<-[:WRITES]-(a:Author)-[:WRITES]->(b2:Book)<-[:REVIEWS]-(p2:User) WHERE id(p1) < id(p2) RETURN id(p1) AS source , id(p2) AS target)

**# Graph Algorithm - Closeness Centrality**

CALL gds.alpha.closeness.stream('user-to-author-relationship')

YIELD nodeId, centrality

RETURN gds.util.asNode(nodeId).user\_id AS user, centrality

ORDER BY user DESC

LIMIT 10