Queries

Basic Queries

Load CSV

```
LOAD CSV WITH HEADERS FROM 'file:///CompleteDatasetTrimmed.csv' AS row FIELD
TERMINATOR ';' WITH row
MERGE (p:Player {
   name: row.Name,
   age: toInteger(row.Age),
   overall: toInteger(row.Overall),
   value: row.Value,
   wage: row.Wage,
   acceleration: toInteger(row.Acceleration),
   aggression: toInteger(row.Aggression),
   agility: toInteger(row.Agility),
   balance: toInteger(row.Balance),
   ballControl: toInteger(row.Ball_control),
   composure: toInteger(row.Composure),
   crossing: toInteger(row.Crossing),
   curve: toInteger(row.Curve),
   dribbling: toInteger(row.Dribbling),
   finishing: toInteger(row.Finishing),
   freeKick: toInteger(row.Free_kick_accuracy),
   gkDiving: toInteger(row.GK_diving),
   gkHandling: toInteger(row.GK_handling),
   gkKicking: toInteger(row.GK_kicking),
   gkPositioning: toInteger(row.GK_positioning),
   gkReflexes: toInteger(row.GK_reflexes),
   headingAcc: toInteger(row.Heading_accuracy),
   interceptions: toInteger(row.Interceptions),
   jumping: toInteger(row.Jumping),
   longPassing: toInteger(row.Long_passing),
   longShotes: toInteger(row.Long_shots),
   marking: toInteger(row.Marking),
   penalties: toInteger(row.Penalties),
   reactions: toInteger(row.Reactions),
   shortPassing: toInteger(row.Short_passing),
   shotPower: toInteger(row.Shot_power),
   slidingTackle: toInteger(row.Sliding_tackle),
   sprintSpeed: toInteger(row.Sprint_speed),
   stamina: toInteger(row.Stamina),
   standingTackle: toInteger(row.Standing_tackle),
   strength: toInteger(row.Strength),
   vision: toInteger(row.Vision),
   volleys: toInteger(row.Volleys),
   positions:split(trim(row.Preferred_Positions), ' ')})
MERGE (n:Nation {name: row.Nationality})
MERGE (c:Club {name: row.Club})
```

```
// Relationships
MERGE (p)-[r1:IS_FROM]->(n)
MERGE (c)-[r2:OWNS]->(p)-[r3:PLAYS_FOR]->(c)
WITH p MATCH (p) UNWIND p.positions as position
MERGE (pos:Position {name: position})
MERGE (p)-[r3:PLAYS]->(pos)
```

Projections

Projection graph

```
CALL gds.graph.project.cypher(
    //Graph name
    'graph',
    //Node Labels
    'MATCH (s) WHERE s:Player OR s:Position OR s:Nation OR s:Club RETURN id(s) AS i
d, labels(s) AS labels, coalesce(s.strength, 0) AS strength, coalesce(s.marking, 0) AS marking
',
    //Relationship types
    'MATCH (s:Player)-[r]->(t) RETURN id(s) AS source, id(t) AS target, type(r) AS type',
    {validateRelationships:FALSE})
YIELD graphName as graph, nodeQuery, nodeCount AS nodes, relationshipQuery, relations
hipCount AS rels
```

Projection player_CB_graph

```
CALL gds.graph.project.cypher(
    //Graph name
    'player_CB_graph',
    //Node Labels
    'MATCH (s) WHERE s:Player RETURN id(s) AS id, labels(s) AS labels, coalesce(s.streng
th, 0) AS strength, coalesce(s.marking, 0) AS marking',
    //Relationship types
    'MATCH (s:Player)-[r:PLAYS]-
>(t:Position{name: "CB"}) RETURN id(s) AS source, id(t) AS target, type(r) AS type',{valid
ateRelationships:FALSE})
YIELD graphName as graph, nodeQuery, nodeCount AS nodes, relationshipQuery, relations
hipCount AS rels
```

Algorithm-based queries

PageRank- Markus

```
// Alg PageRank graph
CALL gds.pageRank.stream('graph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC
```

Label Propagation- Thomas

```
// Alg LPA graph
CALL gds.labelPropagation.stream('graph')
YIELD nodeId, communityId AS Community
RETURN gds.util.asNode(nodeId).name AS Name, Community
ORDER BY Community, Name
```

K-nearest Neighbour- Rasmus

```
// Alg KNN player_CB_graph
CALL gds.knn.stream('player_CB_graph', {
    topK: 1,
    nodeProperties: ['strength','marking'],
    // The following parameters are set to produce a deterministic result
    randomSeed: 1337,
    concurrency: 1,
    sampleRate: 1.0,
    deltaThreshold: 0.0
})
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).name AS Player1, gds.util.asNode(node1).strength AS p1_
strength, gds.util.asNode(node1).marking AS p1_marking, gds.util.asNode(node2).name AS
Player2, gds.util.asNode(node2).strength AS p2_strength, gds.util.asNode(node2).marking A
S p2_marking, similarity
ORDER BY similarity DESCENDING, Player1, Player2
```