**Cypher Queries**

**S1: List peptide and all the protein it belongs as a cluster**

PROFILE MATCH (pep:Peptide)-[:BELONGS\_TO\_PROTEIN]->(pro:Protein) RETURN pep.id as peptide, collect(pro.name) as Protein;

**S2: List peptide and all protein it belongs and other peptide which shares the same protein in a group**

PROFILE MATCH (pepone:Peptide)-[:BELONGS\_TO\_PROTEIN]->(pro:Protein), (peptwo:Peptide)-[:BELONGS\_TO\_PROTEIN]->(pro:Protein) RETURN collect(pro.name) as proteins,pepone.id, collect(peptwo.id);

**S3: List all the proteins ‘Peptide-10 ‘ belongs to in descending order**

PROFILE MATCH (p{id:'Peptide-10'})--(pro) WITH pro RETURN pro.name ORDER BY pro.name DESC;

**S4: List the peptide which belongs to more than one proteins**

PROFILE MATCH (p:Peptide)-[:BELONGS\_TO\_PROTEIN]->(q:Protein) WITH p,COUNT(q) as numberofprotein WHERE numberofprotein>2 RETURN p.id as peptide, numberofprotein;

**S5: Return name of protein that matches Peptide-49**

PROFILE CALL{ MATCH (p{id:'Peptide-49'})--(pro) RETURN pro} MATCH (pro)<-[:BELONGS\_TO\_PROTEIN]-(p:Peptide) RETURN pro.name as protein,collect(p.id) as peptide;

**S6: Return all peptides, protein and relationships**

PROFILE MATCH (n:Peptide)-[r]- (m:Protein) RETURN n,m,r;

**S7: List all proteins and their count belonging to each peptide**

PROFILE MATCH (p:Peptide)-[:BELONGS\_TO\_PROTEIN]->(q:Protein) WITH count(q) as numberofprotein, collect(q.name) as protein,p RETURN p.id as peptide, numberofprotein,protein;

**S8: List the protein and peptides which belongs to same proteins as Pepetide-2**

PROFILE MATCH(pepone:Peptide)-[:BELONGS\_TO\_PROTEIN]->(peptwo:Protein)<-[:BELONGS\_TO\_PROTEIN]-(pe2:Peptide) WHERE pepone.id= “Peptide-2” RETURN pepone.id as Peptide, collect(peptwo.id) as sharedproteins;

**S9: Protein and its list of peptide**

PROFILE MATCH(p:Protein)<-[:BELONGS\_TO\_PROTEIN]-(pep:Peptide) RETURN p.name, collect(pep.id) as `Peptide id`;

**S10: List peptides belonging to Protein**

PROFILE MATCH p=(pep:Peptide)-[r:BELONGS\_TO\_PROTEIN]->(pro:Protein) RETURN p;

**D1:Querying all relations between peptides and proteins**

PROFILE MATCH p=(n:Peptide)-[\*1]-(m:Protein) RETURN p;

**D2: Querying all shared peptide sets**

PROFILE MATCH (n:Peptide)-[r]->(m:Protein)-[:BELONGS\_TO\_PROTEIN]-(n2:Peptide)-[]-(n3:Protein) RETURN n,r,m,n2;

**D3: Querying all shared peptides**

PROFILE MATCH (n:Peptide)-[r]->(m:Protein)-[:BELONGS\_TO\_PROTEIN]-(n2:Peptide) RETURN n,r,m,n2;

**D4: Querying all shared peptides**

PROFILE MATCH p=(n:Peptide)-[r]->(m:Protein)-[:BELONGS\_TO\_PROTEIN]-(n2:Peptide) RETURN p;

**D5: Querying all paths of length 3 that make up peptides and proteins**

PROFILE MATCH p=(n:Peptide)-[\*3]-(m:Protein) RETURN p;

**D6: Querying all paths of shared peptides sets**

PROFILE MATCH p=(n:Peptide)-[r]->(m:Protein)-[:BELONGS\_TO\_PROTEIN]-(n2:Peptide)-[]-(n3:Protein) RETURN p;

**D7: Return all the unique keys of peptide and protein nodes**

PROFILE MATCH (n:Peptide) WITH n UNWIND keys(n) as key RETURN "peptide" as label, collect(distinct key) as keys UNION MATCH (n:Protein) WITH n UNWIND keys(n) as key RETURN "Protein" as label, collect(distinct key) as keys;

**D8: Query all the peptide and protein nodes and return the count of proteins that every peptide is connected to**

PROFILE MATCH (Peptide)-[:BELONGS\_TO\_PROTEIN]->(Protein) RETURN Peptide, count(DISTINCT Protein) AS ProteinCount ORDER BY ProteinCount;

**D9: Return the top 5 peptides that are connected to most proteins**

PROFILE MATCH (Peptide)-[:BELONGS\_TO\_PROTEIN]->(Protein) RETURN Peptide, count(DISTINCT Protein) AS ProteinCount ORDER BY ProteinCount DESC LIMIT 5;

**D10: Return all the distinct sets of shared peptides**

PROFILE MATCH

(Protein1)<-[:BELONGS\_TO\_PROTEIN]-(Peptide1),

(Protein1)<-[:BELONGS\_TO\_PROTEIN]-(Peptide2),

(Protein2)<-[:BELONGS\_TO\_PROTEIN]-(Peptide2),

(Protein2)<-[:BELONGS\_TO\_PROTEIN]-(Peptide1)

WHERE id(Peptide1) <> id(Peptide2) AND id(Protein1) <> id(Protein2)

RETURN DISTINCT Peptide1, Peptide2;

**Q1: Count all nodes**

PROFILE MATCH (n)

RETURN count(n);

**Q2: Count all relationships**

PROFILE MATCH ()-->() RETURN count(\*);

**Q3: What kind of nodes exist. Sample some nodes, reporting on property and relationship counts per node.**

PROFILE MATCH (n) WHERE rand() <= 0.1 RETURN DISTINCT labels(n), count(\*) AS SampleSize, avg(size(keys(n))) as Avg\_PropertyCount, min(size(keys(n))) as Min\_PropertyCount, max(size(keys(n))) as Max\_PropertyCount, avg(size( (n)-[]-() ) ) as Avg\_RelationshipCount,

min(size( (n)-[]-() ) ) as Min\_RelationshipCount, max(size( (n)-[]-() ) ) as Max\_RelationshipCount;

**Q4: Get Proteins of specific metaprotein**

PROFILE MATCH(n) WHERE n.metaprotein ='7' RETURN (n);

**Q5: Group all the peptides together with respective proteins and metaproteins**

PROFILE MATCH(Peptide:Peptide)-[:BELONGS\_TO\_PROTEIN]->(Protein:Protein)

RETURN Protein.name AS Proteins, Collect(Peptide.id) AS Peptides, Protein.metaprotein AS Metaprotein;

**GDS:**

**P1: GDS Create subgraph algorithm**

PROFILE CALL gds.graph.create( 'myGraph',{Peptide:{label:'Peptide'}, Protein:{label:'Protein'} }, '\*')

YIELD graphName, nodeCount, relationshipCount;

**P2: GDS Node Similarity Algorithm**

PROFILE CALL gds.nodeSimilarity.stream('myGraph')

YIELD node1, node2, similarity

RETURN gds.util.asNode(node1).name AS Peptide1, gds.util.asNode(node2).name AS Peptide2, similarity

ORDER BY similarity DESCENDING, Peptide1, Peptide2;

**P3: Fast Random Projection**

PROFILE CALL gds.fastRP.stream.estimate('myGraph', {embeddingDimension: 128})

YIELD nodeCount, relationshipCount, bytesMin, bytesMax, requiredMemory;

**P4: Page rank**

PROFILE CALL gds.pageRank.write.estimate('myGraph', {

writeProperty: 'id',

maxIterations: 5,

dampingFactor: 0.85

})

YIELD nodeCount, relationshipCount, bytesMin, bytesMax, requiredMemory;

**P5: List all GDS graphs and their names**

PROFILE CALL gds.graph.list()

YIELD graphName;

**Apoc:**

**P1: Apoc connected nodes**

PROFILE MATCH (p1:Protein {id:"Protein-156"})

MATCH (p2:Peptide {id:"Peptide-7"})

RETURN apoc.nodes.connected(p1, p2, "BELONGS\_TO\_PROTEIN") AS output;

**P2: Diff b/w 02 nodes**

PROFILE MATCH (p1:Protein {id:"Protein-156"})

MATCH (p2:Peptide {id:"Peptide-7"})

RETURN apoc.diff.nodes(p1, p2) AS output;

**P3: Add labels to nodes**

PROFILE CREATE (:Protein {title: 'Speices', genre: 'label'});

MATCH (p:Protein)

CALL apoc.create.addLabels( p, [ p.label ] )

YIELD node

REMOVE node.label

RETURN node limit 25;

**P4: Parallel node search**

PROFILE CALL apoc.search.nodeAll('{Protein:"id", metaprotein: "4"}','exact', "Protein-156") YIELD node AS n RETURN n;

**P5: Neighbour to 1 or 2 hops**

PROFILE MATCH (p:Protein {id: "Protein-156"})

CALL apoc.path.subgraphNodes(p, {

relationshipFilter: "BELONGS\_TO\_PROTEIN",

minLevel: 1,

maxLevel: 2

})

YIELD node

RETURN node;