Connectivity Analytics with CYPHER

```
//Viewing the graph
match (n:MyNode)-[r]->(m)
return n, r, m

// Find the outdegree of all nodes
match (n:MyNode)-[r]->()
return n.Name as Node, count(r) as Outdegree
order by Outdegree
union
match (a:MyNode)-[r]->(leaf)
where not((leaf)-->())
return leaf.Name as Node, 0 as Outdegree

// Find the indegree of all nodes
match (n:MyNode)<-[r]-()
return n.Name as Node, count(r) as Indegree
order by Indegree
union
match (a:MyNode)<-[r]-(root)
where not((root)<--())
return root.Name as Node, 0 as Indegree

// Find the degree of all nodes
match (n:MyNode)-[r]-()
return n.Name, count(distinct r) as degree
order by degree

// Find degree histogram of the graph
match (n:MyNode)-[r]-()
with n as nodes, count(distinct r) as degree
return degree, count(nodes) order by degree asc

//Save the degree of the node as a new node property
match (n:MyNode)-[r]-()
with n, count(distinct r) as degree
set n.deg = degree
return n.Name, n.deg
```

```
// Construct the Adjacency Matrix of the graph
match (n:MyNode), (m:MyNode)
return n.Name, m.Name,
case
when (n)-->(m) then 1
else 0
end as value

// Construct the Normalized Laplacian Matrix of the graph
match (n:MyNode), (m:MyNode)
return n.Name, m.Name,
case
when n.Name = m.Name then 1
when (n)-->(m) then -1/(sqrt(toInteger(n.deg))*sqrt(toInteger(m.deg)))
else 0
end as value
```