```
match (p:Person)-[f:FRIEND_OF]-()
return p,f
limit 50

match (p:Person)-[b:BOUGHT]-(prod:Product)
return p.name, count(prod)
order by count(prod) desc;

//Friendship-based Recommendation
match path=(p1:Person {name:'Lynda Willis'})-[:FRIEND_OF]-(p2:Person)-[:BOUGHT]->(prod:Product)
where not((p1)-[:BOUGHT]-(prod))
return path

//Commonality-based Recommendation
match (p1:Person)-[:HAS_COMPETENCY]-(c:Competency)-[:HAS_COMPETENCY]-(p2:Person)
with p1,p2
match (p1)-->(co:Company)<--(p2)-[:BOUGHT]->(prod:Product)
where not((p1)-[:BOUGHT]-(prod))
return p1.name, p2.name, prod.name

//Friendship Gap
match path=(p1:Person)-[:FRIEND_OF*2..2]-(p2:Person)
where not((p1)-[:FRIEND_OF]-(p2))
return path
limit 10

//Friendship recommendation
match path=(p1:Person)-[:HAS_COMPETENCY]-(c:Competency)-[:HAS_COMPETENCY]-(p2:Person)
with path,p1,p2
match (p1)-->(co:Company)<--(p2)
where not((p1)-[:FRIEND_OF]-(p2))
return path
limit 10

//Degree Centrality
match (n:Person)-[r:FRIEND_OF]->(m:Person)
return n.name, count(r) as DegreeCentrality
order by DegreeCentrality desc

//Betweenness Centrality
match p=allShortestPaths((n:Person)-[:FRIEND_OF*]-(m:Person))
where id(n)<id(m) and length(p)>1
unwind nodes(p)[1..-1] as allnodes
return allnodes.name, count(*) as BetweennessCentrality
order by BetweennessCentrality desc
```