Q1) Visualiser le graphe (afficher tous les noeuds et tous les arcs). Quelles sont les propriétés et les étiquettes des noeuds et des arcs?

```
match (n) return n
```

Q2) Ajouter une étiquette :PERSONNAGE aux noeuds qui ont une propriété "personnageid" (utiliser set et exists).

```
match (n)
where exists(n.personnageid)
SET n:PERSONNAGE
return distinct labels(n)
```

Q3) Compter le nombre de noeuds.

```
match (n) return count(n)
Ou
match (n) return count(*)
Ou
match () return count(*)
```

Q4) Compter le nombre d'arcs.

```
match ()-[r]->() return count(*)
Ou
match ()-->() return count(*)
```

Q5) Trouver l'étiquette du noeud dont le nom est 'Jules Cesar'.

```
match (n{name:"Jules Cesar"}) return labels(n)
```

Q6) Trouver l'étiquette des arcs vers le noeud dont le nom est 'Le Domaine des dieux'.

```
MATCH (n {name:"Le Domaine des dieux"}) <- [r ] - ()
RETURN TYPE(r)
```

```
Réponse: "APPARAIT_DANS"
```

Q7) Afficher toutes les propriétés de n'importe quels trois noeuds parmi les noeuds de type PERSONNAGE.

```
match (n:PERSONNAGE) return properties(n) limit 3
```

Q8) Afficher les noms des personnages qui n'ont pas d'arc sortant dont le type est COMPAGNON_AVENTURE.

```
match (n:PERSONNAGE)
where Not exists((n)-[:COMPAGNON_AVENTURE]->())
return n.name
```

Q9) Calculer les triangles du graphe et afficher seulement deux parmi ces triangles (pour chacun de ces deux triangles afficher les noms des noeuds correspondants). On considère le graphe comme étant non-dirigé.

```
match (a)--(b)--(c)
match (c)--(a)
return a.name, b.name, c.name limit 2
```

***Pas la même réponse -> cmt faire triangle ?

Q10) Afficher les noms des trois couples de personnages qui apparaissent dans le même album, pour chaque couple de personnages affichez l'id de album commun.

match (p1:PERSONNAGE) - [r1:APPARAIT_DANS] -> (a)
match (p2:PERSONNAGE) - [r2:APPARAIT_DANS] -> (a)
where exists(a.albumid) and p1<p2
return p1.name, p2.name, a.albumid limit 3
***Réponse: pas idem mais peut-être bon ?

Q11) Afficher le sous-graphe des personnages reliés par des arcs de type COMPAGNON_AVENTURE qui ne contient pas 'Cleopatre'.

Q12) Afficher le sous-graphe contenant comme noeuds source 'Cleopatre', 'Jules Cesar' ou 'Caius Obtus' et comme noeuds destination 'Cesarion (Ptolemee XVI)' ou 'Caligula Alavacomgetepus'. Les noeuds sont reliés par des arcs de type COMPAGNON_AVENTURE.

```
match (s:PERSONNAGE) - [r:COMPAGNON_AVENTURE] -> (d:PERSONNAGE)
where (s.name="Cleopatre" or s.name="Jules Cesar" or s.name="Caius Obtus") and
(d.name="Cesarion (Ptolemee XVI)" or d.name="Caligula Alavacomgetepus")
```

```
return s.personnageid, d.personnageid
```

Q13) Afficher le sous-graphe contenant les noeuds qui se trouvent à une distance 3 de 'Cleopatre' ainsi que les arcs de type COMPAGNON_AVENTURE qui les relient. On considère uniquement les arcs dirigés de type COMPAGNON_AVENTURE.

match (c {name: 'Cleopatre'}) - [r:COMPAGNON_AVENTURE*3] - (n:PERSONNAGE)
return distinct n.name
Neo.ClientError.Statement.SyntaxError

match (c {name: 'Cleopatre'}) - [r:COMPAGNON_AVENTURE*3] - (n)
return n
Ou
MATCH (c) -[:COMPAGNON_AVENTURE*3]-(n:PERSONNAGE)
WHERE c.name = 'Cleopatre' and NOT (c)-[:COMPAGNON_AVENTURE]-(n)
RETURN n
***aucune bonne réponse

Q14) Afficher le plus court chemin et sa longueur entre Jules Cesar et Epidemais.

MATCH p=shortestPath((j)-[r *1..10]-(e))
WHERE j.name="Jules Cesar" and e.name = "Epidemais"
RETURN EXTRACT(n in node(p) | labels(n)), type(r), length(p)
*** Msg erreur : Extract is no longer supported.

Q15) Afficher le plus long chemin de type COMPAGNON_AVENTURE et sa longueur entre Jules Cesar et Brutus (graphe non-dirigé).

```
match p=((a)-[:COMPAGNON_AVENTURE*1..10]-(b))
where a.name="Jules Cesar" and b.name="Brutus"
return p, length(p) as d order by d desc limit 1
```

***pas sûr d'être la bonne réponse

Q16) Afficher les 3 noeuds qui ont le degré le plus élevé. Affichez pour chacun de ces noeuds son nom et son degré. Pour calculer le degré considérer uniquement les arcs de type NATIONALITE ou PERSONNAGE_TYPE.

```
match (a)-[:PERSONNAGE_TYPE|NATIONALITE]-(b)
with a, count(b) as con
return a.name, con order by con desc limit 3
```

***ok

Q17) Afficher pour chaque couple de personnages reliés par un arc (non-dirigé) de type COMPAGNON_AVENTURE le nombre d'albums qu'ils ont en commun (afficher le triplet contenant les noms des personnages et le nombre d'albums en commun). Chaque couple de personnages doit apparaître une seule fois. Trier par nombre total d'albums décroissant.

```
match (a:PERSONNAGE)-[:APPARAIT_DANS]-(c)
match (b:PERSONNAGE)-[:APPARAIT_DANS]-(c)
where a.personnageid < b.personnageid
with a, b, count(c) as commun
match (a)-[:COMPAGNON_AVENTURE]-(b)
return distinct a.name, b.name, commun
order by commun desc
```

***ok

Q18) Modifier la requête précédente afin d'ajouter à l'arc de type COMPAGNON_AVENTURE qui relie les personnages retournés une propriété 'albums' dont la valeur est le nombre total d'albums en commun. Retourner les noms des personnages ainsi que la propriété ajoutée (utiliser WITH pour enchaîner les opérations).

```
match (a:Personnage)-[:APPARAIT_DANS]->(c:Album)
match (b:Personnage)-[:APPARAIT_DANS]->(c:Album)
match (a)-[r:COMPAGNON_AVENTURE]-(b)
where a.personnageid < b.personnageid
with a, b, r, count(c) as con
set r.album = con
```

*** ????

Q19) Degré total des noeuds

Afficher pour chaque valeur de degré le nombre de noeuds avec ce degré. Considérer tous les arcs, ordonner par degré.

```
match (a)--(b) with a, count(b) as con return con, count(a) as n order by con
```

***ok

Pour chaque noeud enregistrer son degré comme nouvelle propriété. Affichez les 3 premières noeuds avec le degré le plus élevé.

```
match (a)--(b)
with a, count(b) as con
set a.degree = con
return a.name, a.degree
order by a.degree desc limit 3
```

***ok

Q20)Afficher pour chaque noeud son nom et son degré sortant. Pour les noeuds sans liens sortants afficher 0. Ordonner par ordre décroissant des degrés. Considérer uniquement les arcs de type NATIONALITE et PERSONNAGE_TYPE.

```
match (a)-[:PERSONNAGE_TYPE|NATIONALITE]->(b)
return a.name as nom, count(b) as degreSortant
union all
match (c)
where not (c)-[:PERSONNAGE_TYPE|NATIONALITE]->()
return c.name as nom, 0 as degreSortant
```

Q21) Afficher les noms des personnages sur le plus court chemin de type COMPAGNON_AVENTURE entre Numerobis et Assurancetourix (utiliser la fonction EXTRACT) (graphe non-dirigé).

```
match p=shortestPath((a)-[:COMPAGNON_AVENTURE*1..10]-(b)) where a.name="Numerobis" and
b.name="Assurancetourix" return [ n in nodes(p) | n.name ]
```

*** return EXTRACT (n in nodes(p) l n.name)

Q22) Afficher le diamètre du graphe en considérant uniquement les noeuds de type PERSONNAGE et les arcs de type COMPAGNON_AVENTURE

```
match p=shortestPath((a:Personnage)-[:COMPAGNON_AVENTURE*1..10]-(b:Personnage))
where a.name <> b.name
return max(length(p))
```

Q23) Afficher les noms des noeuds sur tous les plus courts chemins entre Numerobis et Assurancetourix. Afficher uniquement les plus courts chemins contenant plus de 4 noeuds (utiliser la fonction EXTRACT)(graphe non-dirigé).

```
match p=allShortestPaths((a:Personnage)-[:COMPAGNON_AVENTURE*1..10]-(b:Personnage))
where a.name = "Numerobis" and b.name="Assurancetourix" and length(p)>=4
return [n in nodes(p) | n.name]
```

Q24) On considère la propriété 'albums' des arcs de type COMPAGNON_AVENTURE comme poids des arcs. Afficher le chemin avec la distance la plus courte entre Numerobis et Assurancetourix qui considère ces poids (même résultat que celui obtenu avec l'algorithme de Dijkstra). Considérer uniquement les arcs de type COMPAGNON_AVENTURE et leur poids (aide (utiliser shortestPath… WITH REDUCE): WITH REDUCE(dist = 0, rel in rels(path) l dist+rel.albums)).

```
match p=shortestPath((a:Personnage {name:"Numerobis"})-[:COMPAGNON_AVENTURE*1..10]-
(b:Personnage {name: "Assurancetourix"}))
with p, reduce(dist = 0, rel in relationships(p) | dist+rel.album) as dist
return p, dist
```

Q25) Effacer tous les noeuds et leur arcs

Q26) Effacer tous les nodes qui n'ont pas d'arcs