# Exercice 1

## 1. Lister les 200 premières données de la base (nœuds et leur relations)

```
match (n)-[r]-() return n, r limit 200
```

Cette requête trouve 200 nœuds et relations.

## 2. Lister tous les users

```
match (n:User) return n
```

Cette requête recupère tous les nœuds user.

## 3. Donner l'âge de l'utilisateur dont l'id est 5

```
match (n:User {id: 5}) return n.age
```

```
33
```

Cette requête recupère le user 5 et renvoie son âge.

## 4. Afficher les 20 premiers films

```
match (n:Movie) return n limit 20
```

Cette requête recupère et renvoie 20 films.

## 5. Afficher les films qui ont été notés par l'utilisateur dont l'id est 1

```
match (:User {id: 1})-->(m:Movie) return m
```

Cette requête recupère le user 1 puis tous les films qu'il a noté.

## 6. Afficher les films qui ont été notés par l'utilisateur dont l'id est 1 ainsi que leur genres

```
match (:User {id: 1})-->(m:Movie)-->(g:Genre) return m.title, g.name
```

Cette requête recupère le user 1 puis tous les films qu'il a noté et ensuite le genre de ces films.

## 7. Afficher les notes des films de genre « Drama »

```
match ()-[r:RATED]->(m:Movie)-->(:Genre {name: "Drama"})
return m.title, avg(r.rating)
```

Cette requête recupère toutes les notations sur les films dont le genre est `Drama`.

## 8. Lister les utilisateurs de sexe féminin qui sont soit écrivaines soit artistes

```
match (u:User {sex: "F"})
where u.occupation = "writer" OR u.occupation = "artist"
return u
```

Cette requête recupère tous les User avec `sex: "F"` puis filtre pour garder seulement les écrivaines et les artistes.

## 9. Quel est l'âge moyen des étudiants ?

```
match (u:User {occupation: "student"}) return avg(u.age)
```

```
23.193548387096776
```

Cette requête recupère tous les users étudiants puis renvoie la moyenne de leurs âges.

## 10. Quel est l'âge moyen par occupation ?

```
match (u:User) return u.occupation, avg(u.age)
```

Cette requête aggrège les users par occupation et fait la moyenne des âges par occupation.

## 11. Quelles sont les 3 « occupations » les plus populaires ?

```
match (u:User)
with u.occupation as occupation, count(u) as nb
return occupation, nb
order by nb desc limit 3
```

```
"student"       31
"administrator" 23
"other"         19
```

Cette requête aggrège les occupations et compte le nombre d'utilisateurs par occupation, puis affiche les 3 premiers résultats triés par nombre d'utilisateurs.

## 12. Combien de valeurs différentes existe-t-il pour l'attribut occupation ?

```
match (u:User) return count(DISTINCT u.occupation)
```

```
21
```

Cette requête compte le nombre d'occupations différentes

## 13. Quels sont les films produits en 1995 ? (la date de production est contenue dans le titre du film)

```
match (m:Movie)
where m.title ends with "(1995)"
return m.title
```

Cette requête récupère les films dont le titre se termine par `(1995)`, ce qui correspond aux films sortis en 1995.

## 14. Ajouter pour chaque film la propriété date.

```
match (m:Movie)
SET m.date = toInteger(left(right(m.title, 5), 4))
```

Cette requête récupère les 5 derniers caractères du titre d'un film (l'année et une parenthèse), puis les 4 premiers caractères de cette sélection (l'année seule), convertit le résultat en entier, puis l'affecte à la propriété date.

## 15. Lister les relations avec leur nombres d'occurrences

```
match ()-[r]-()
return type(r), count(r)
```

```
"FRIEND_OF"       1600
"RATED"          22620
"CATEGORIZED_AS"  1668
```

Cette requête aggrège les relations par type et compte le nombre de relations par type.

## 16. Quels sont les 5 films les plus notés

```
match ()-[r:RATED]->(m:Movie)
with m.title as movie, count(r) as nb_ratings
return movie, nb_ratings
ORDER BY nb_ratings desc
limit 3
```

```
"Star Wars (1977)" 118
"Contact (1997)"   116
"Fargo (1996)"     105
"Liar Liar (1997)" 102
"English Patient"  100
```

Cette requête recupère toutes les notations, les aggrège par film, et compte le nombre de notations par film.

## 17. Combien de films ont reçu au moins une fois la note 1

```
match ()-[r:RATED {rating: 1}]->(m:Movie)
return count(DISTINCT m)
```

```
266
```

Cette requête recupère toutes les notations à 1 ainsi que leurs films et compte les nombre de films distincts.

## 18. Donner la liste des films qui ont reçu au moins une fois la note 1 en donnant le nombre de fois où ils ont reçu cette note et qui a donné la note.

```
match (u: User)-[r:RATED {rating: 1}]->(m:Movie)
return m.title, count(r), collect(u.id)
```

Cette requête recupère toutes les notations à 1 ainsi que leurs films et qui a mis les notations, puis aggrège les résultats par film, et affiche le nombre de notations ainsi que la liste des utilisateurs ayant donné cette notation à ce film.

## 19. Quels sont les films qui ont une note moyenne >4

```
match ()-[r:RATED]->(m:Movie)
with avg(r.rating) as avg_rating, m.title as movie
where avg_rating > 4
return movie, avg_rating
```

Cette requête recupère toutes les notations ainsi que leurs films, aggrège les résultats par films, calcule la note moyenne par film, et filtre sur la note moyenne.

## 20. Quels sont les films regardés par le user 13

```
match (:User {id: 13})-->(m:Movie)
return m.title
```

Cette requête récupère le user 13 et tous les films avec lesquels il a un lien.

## 21. Quels sont les films non regardés par le user 13 ?

```
match (u:User)-->(m:Movie)
with m, COLLECT(u.id) as watchers
where not 13 in watchers
return m.title
```

Cette requête recupère tous les liens entre user et films, aggrège par film, fait la liste des utilisateurs ayant regardé chaque film, et filtre pour ne garder que les films que le user 13 n'a pas noté.

## 22. Donner le nombre des connexions réflexive

```
match (n)-[r]->(n)
return count(r)
```

```
0
```

Cette requête recupère toutes les relations réflexives

## 23. Donner le nombre des connexions non-réflexives

```
match (n)-[r]->(m)
where not n = m
return count(r)
```

```
12944
```

Cette requête recupère toutes les relations non-réflexives

## 24. Lister les amis et les amis des amis du user 1, vous pouvez utiliser un pattern de chemin de longueur 2. https://neo4j.com/docs/cypher-manual/current/patterns/

```

```

```
match ((u: User)-->(f: User)){1,2}
where u[0].id = 1
return distinct f[-1].id as friend_id
```

Cette requête recupère les relations entre utilisateurs de longueur 1 ou 2, filtre pour garder celles commençant par le user 1, et renvoie les valeurs distinctes du user en bout de chaîne.

## 25. Lister les amis et les amis des amis du user 1 en donnant leur distance du user 1 (cette distance est 1 lorsque c'est un ami direct, 2 lorsque c'est un ami d'un ami).

```
match ((u: User)-->(f: User)){1,2}
where u[0].id = 1
return distinct f[-1].id as friend_id, min(size(f)) as depth
```

Cette requête recupère les relations entre utilisateurs de longueur 1 ou 2, filtre pour garder celles commençant par le user 1, aggrège par id de user ami, puis recupère la longueur de la chaîne la plus courte pour y arriver, et renvoie l'id du user ainsi que la longueur de la chaîne.

## 26. Quel est l'utilisateur qui a le plus d'amis en communs avec l'utilisateur 41

```
match (o: User)--(c:User)--(u:User {id: 41})
where not o = u
with count(distinct c) as common, o as other
return other.id, common
order by common desc
limit 1
```

Cette requête recupère tous les amis d'amis du user 41, aggrège par ami d'ami, compte le nombre d'amis en commun, et renvoie l'ami d'ami avec le plus d'amis en commun.

## 27. Quel utilisateur a noté le plus grand nombre de films que l'utilisateur 1 a également notés et quel est ce nombre ?

```
match (u: User)-->(m:Movie)<--(:User {id: 1})
with count(m) as nb_movies, u
return u.id, nb_movies
order by nb_movies desc
limit 1
```

Cette requête recupère tous les films notés par l'utilisateur 1 ainsi que toutes les personnes ayant noté ces films, aggrège par utilisateur, compte le nombre de films en communs, et renvoie l'utilisateur avec le plus de films en commun.

## Bonus: Compter les films sans l'année dans le titre

```
match (m:Movie)
where m.title ends with ")"
return count(m.title)
```

```
92
```

Cette requête trouve tous les films dont le titre ne termine pas par une parenthèse, donc n'ayant pas l'année à la fin du titre.