

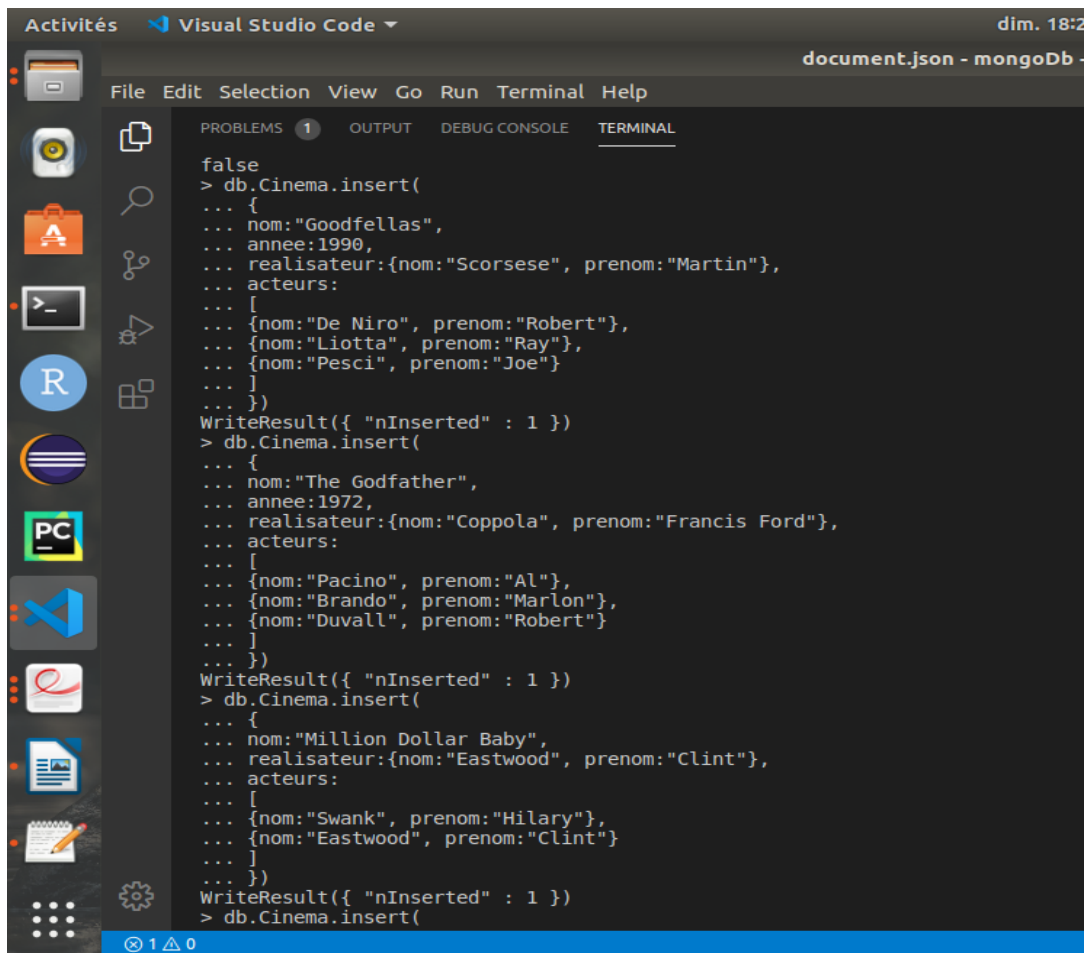
## Table des matières

RENDU TRAVAUX PRATIQUES SUR LES BASES DE DONNEES NON RELATIONNELLES .....	2
TRAVAUX PRATIQUE I (MongoDB) .....	3
1. Question 1 .....	3
2. Question 2 .....	3
3. Question 3 .....	4
4. Question 4 .....	4
5. Question 5 .....	5
6. Question 6 .....	5
7. Question 7 .....	6
8. Question 8 .....	6
9. Question 9 .....	6
10. Question 10 .....	7
11. Question 11 .....	8
12. Question 12 .....	9
13. Question 13 .....	9
14. Question 14 .....	10
15. Question 15 .....	11
16. Question 16 .....	11
TRAVAUX PRATIQUE I (NEO4j) .....	12
1. Question 1 .....	12
2. Question 2 .....	12
3. Question 3 .....	12
4. Question 4 .....	13
5. Question 5 .....	14
6. Question 6 .....	14
7. Question 7 .....	14
8. Question 8 .....	15
9. Question 9 .....	16

## RENDU TRAVAUX PRATIQUES SUR LES BASES DE DONNEES NON RELATIONNELLES

# TRAVAUX PRATIQUE I (MongoDB)

## 1. Question 1



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the following commands and output:

```
false
> db.Cinema.insert(
... {
...   nom:"Goodfellas",
...   annee:1990,
...   realiseur:{nom:"Scorsese", prenom:"Martin"},
...   acteurs:
...   [
...     {nom:"De Niro", prenom:"Robert"},
...     {nom:"Liotta", prenom:"Ray"},
...     {nom:"Pesci", prenom:"Joe"}
...   ]
... })
WriteResult({ "nInserted" : 1 })
> db.Cinema.insert(
... {
...   nom:"The Godfather",
...   annee:1972,
...   realiseur:{nom:"Coppola", prenom:"Francis Ford"},
...   acteurs:
...   [
...     {nom:"Pacino", prenom:"Al"},
...     {nom:"Brando", prenom:"Marlon"},
...     {nom:"Duvall", prenom:"Robert"}
...   ]
... })
WriteResult({ "nInserted" : 1 })
> db.Cinema.insert(
... {
...   nom:"Million Dollar Baby",
...   realiseur:{nom:"Eastwood", prenom:"Clint"},
...   acteurs:
...   [
...     {nom:"Swank", prenom:"Hilary"},
...     {nom:"Eastwood", prenom:"Clint"}
...   ]
... })
WriteResult({ "nInserted" : 1 })
> db.Cinema.insert(
```

## 2. Question 2

```
db.Cinema.find({"annee":1990})
```

```

> db.Cinema.find({'annee' : 1990})
{ "id" : ObjectId("5e9c96fb591d10dac6642505"), "nom" : "Goodfellas", "annee" : 1990, "realisateur" : { "nom" : "Scorsese", "prenom" : "Martin" }, "acteurs" : [ { "nom" : "De Niro", "prenom" : "Robert" }, { "nom" : "Liotta", "prenom" : "Ray" }, { "nom" : "Pesci", "prenom" : "Joe" } ] }

```

### 3. Question 3

```
db.Cinema.find({"annee":{$lt:2000}})
```

```

> db.Cinema.find({annee : {$lt : 2000}})
{ "id" : ObjectId("5e9c96fb591d10dac6642505"), "nom" : "Goodfellas", "annee" : 1990, "realisateur" : { "nom" : "Scorsese", "prenom" : "Martin" }, "acteurs" : [ { "nom" : "De Niro", "prenom" : "Robert" }, { "nom" : "Liotta", "prenom" : "Ray" }, { "nom" : "Pesci", "prenom" : "Joe" } ] }
{ "id" : ObjectId("5e9c96fc591d10dac6642506"), "nom" : "The Godfather", "annee" : 1972, "realisateur" : { "nom" : "Coppola", "prenom" : "Francis Ford" }, "acteurs" : [ { "nom" : "Pacino", "prenom" : "Al" }, { "nom" : "Brando", "prenom" : "Marlon" }, { "nom" : "Duvall", "prenom" : "Robert" } ] }
{ "id" : ObjectId("5e9c96fc591d10dac664250b"), "nom" : "Honkytonk Man", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "annee" : 1982, "acteurs" : [ { "nom" : "Eastwood", "prenom" : "Kyle" }, { "nom" : "Bloom", "prenom" : "Verna" } ] }
> db.Cinema.find({annee : {$lt : 2000}}).count()
3

```

### 4. Question 4

```
db.Cinema.find({"realisateur":{"nom":"Eastwood",
"prenom":"Clint"}})
```

```

>
> db.Cinema.find({realisateur : { 'nom' : 'Eastwood', 'prenom' : 'Clint'}}).count()
5
> db.Cinema.find({realisateur : { 'nom' : 'Eastwood', 'prenom' : 'Clint'}})
{ "_id" : ObjectId("5e9c96fc591d10dac6642507"), "nom" : "Million Dollar Baby", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Swank", "prenom" : "Hilary" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac6642508"), "nom" : "Gran Torino", "annee" : 2008, "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Vang", "prenom" : "Bee" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac6642509"), "nom" : "Unforgiven", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Hackman", "prenom" : "Gene" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac664250a"), "nom" : "Mystic River", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Penn", "prenom" : "Sean" }, { "nom" : "Bacon", "prenom" : "Kevin" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac664250b"), "nom" : "Honkytonk Man", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "annee" : 1982, "acteurs" : [ { "nom" : "Eastwood", "prenom" : "Kyle" }, { "nom" : "Bloom", "prenom" : "Verna" } ] }
> db.Cinema.find({realisateur : { 'nom' : 'Eastwood', 'prenom' : 'Clint'}}).count()
5
>

```

## 5. Question 5

db.Cinema.find({"realisateur.prenom":"Clint"})

```

>
>
>
>
> db.Cinema.find({'realisateur.prenom' : 'Clint'})
{ "_id" : ObjectId("5e9c96fc591d10dac6642507"), "nom" : "Million Dollar Baby", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Swank", "prenom" : "Hilary" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac6642508"), "nom" : "Gran Torino", "annee" : 2008, "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Vang", "prenom" : "Bee" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac6642509"), "nom" : "Unforgiven", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Hackman", "prenom" : "Gene" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac664250a"), "nom" : "Mystic River", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Penn", "prenom" : "Sean" }, { "nom" : "Bacon", "prenom" : "Kevin" } ] }
{ "_id" : ObjectId("5e9c96fc591d10dac664250b"), "nom" : "Honkytonk Man", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "annee" : 1982, "acteurs" : [ { "nom" : "Eastwood", "prenom" : "Kyle" }, { "nom" : "Bloom", "prenom" : "Verna" } ] }
>
>
>
>

```

## 6. Question 6

db.Cinema.find({"realisateur.prenom":"Clint", "annee":{"\$lt:2000}})

```

>
>
>
> db.Cinema.find({'realisateur.prenom' : 'Clint', 'annee' : {$lt : 2000}})
{ "_id" : ObjectId("5e9c96fc591d10dac664250b"), "nom" : "Honkytonk Man", "realisateur" : { "nom" : "Eastwood", "prenom" : "Clint" }, "annee" : 1982, "acteurs" : [ { "nom" : "Eastwood", "prenom" : "Kyle" }, { "nom" : "Bloom", "prenom" : "Verna" } ] }
> db.Cinema.find({'realisateur.prenom' : 'Clint', 'annee' : {$lt : 2000}}).count()
1
>
>
>
>

```

## 7. Question 7

```
db.Cinema.find({"acteurs":{"nom":"Eastwood",  
prenom:"Clint"}})
```

```
>  
>  
>  
> db.Cinema.find({ acteurs : {'nom': 'Eastwood', 'prenom' : 'Clint' } })  
{ "_id" : ObjectId("5e9c96fc591d10dac6642507"), "nom" : "Million Dollar Baby", "realisateur" : { "nom" : "Eastwood", "pre  
nom" : "Clint" }, "acteurs" : [ { "nom" : "Swank", "prenom" : "Hilary" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }  
{ "_id" : ObjectId("5e9c96fc591d10dac6642508"), "nom" : "Gran Torino", "annee" : 2008, "realisateur" : { "nom" : "Eastwood  
, "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Vang", "prenom" : "Bee" }, { "nom" : "Eastwood", "prenom" : "Clint" } ]  
}  
{ "_id" : ObjectId("5e9c96fc591d10dac6642509"), "nom" : "Unforgiven", "realisateur" : { "nom" : "Eastwood", "prenom" : "Cl  
int" }, "acteurs" : [ { "nom" : "Hackman", "prenom" : "Gene" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }  
>  
>  
>
```

## 8. Question 8

```
db.Cinema.find({"acteurs":{"$elemMatch":{"nom":"Eastw  
ood"}}}))
```

```
>  
>  
> db.Cinema.find({ acteurs : { $elemMatch : {'nom': 'Eastwood' } } })  
{ "_id" : ObjectId("5e9c96fc591d10dac6642507"), "nom" : "Million Dollar Baby", "realisateur" : { "nom" : "Eastwood", "pre  
nom" : "Clint" }, "acteurs" : [ { "nom" : "Swank", "prenom" : "Hilary" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }  
{ "_id" : ObjectId("5e9c96fc591d10dac6642508"), "nom" : "Gran Torino", "annee" : 2008, "realisateur" : { "nom" : "Eastwood  
, "prenom" : "Clint" }, "acteurs" : [ { "nom" : "Vang", "prenom" : "Bee" }, { "nom" : "Eastwood", "prenom" : "Clint" } ]  
}  
{ "_id" : ObjectId("5e9c96fc591d10dac6642509"), "nom" : "Unforgiven", "realisateur" : { "nom" : "Eastwood", "prenom" : "Cl  
int" }, "acteurs" : [ { "nom" : "Hackman", "prenom" : "Gene" }, { "nom" : "Eastwood", "prenom" : "Clint" } ] }  
{ "_id" : ObjectId("5e9c96fc591d10dac664250b"), "nom" : "Honkytonk Man", "realisateur" : { "nom" : "Eastwood", "prenom" :  
"Clint" }, "annee" : 1982, "acteurs" : [ { "nom" : "Eastwood", "prenom" : "Kyle" }, { "nom" : "Bloom", "prenom" : "Verna  
" } ] }  
>  
>  
>
```

## 9. Question 9

```
db.Cinema.find({"acteurs":{"$elemMatch":{"nom":"Eastwood"}}},{"nom":1,"_id":0})
```

```

>
>
>
>
>
> db.Cinema.find({"acteurs" : { $elemMatch : {"nom": 'Eastwood'}}}, {"nom" : 1 })
{ "_id" : ObjectId("5e9c96fc591d10dac6642507"), "nom" : "Million Dollar Baby" }
{ "_id" : ObjectId("5e9c96fc591d10dac6642508"), "nom" : "Gran Torino" }
{ "_id" : ObjectId("5e9c96fc591d10dac6642509"), "nom" : "Unforgiven" }
{ "_id" : ObjectId("5e9c96fc591d10dac664250b"), "nom" : "Honkytonk Man" }
>
>
>
> db.Cinema.find({"acteurs" : { $elemMatch : {"nom": 'Eastwood'}}}, {"nom" : 1 }).count()
4
>
>
>
>
>
>
>
>

```

Ln 38, Col 46 (4)

## 10. Question 10

```
conn = new Mongo();
```

```
db = conn.getDB("mydb");
```

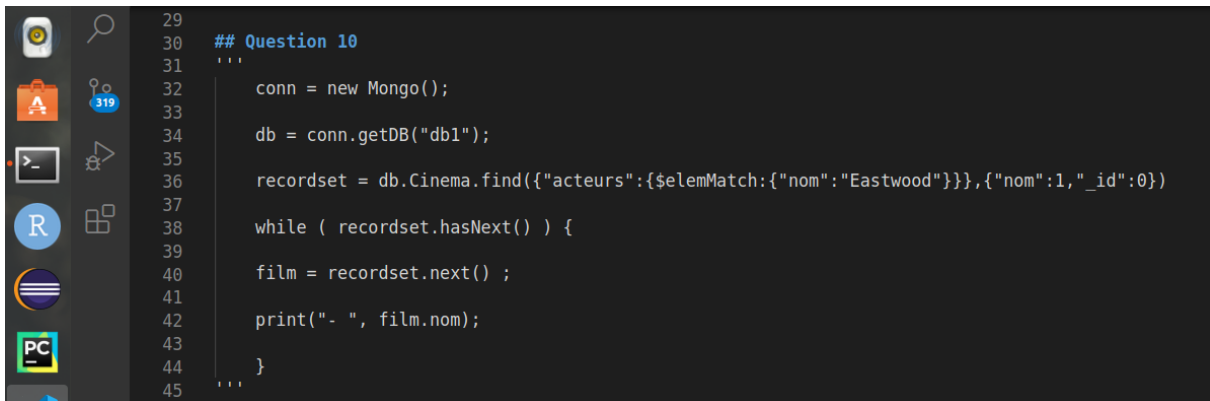
```
recordset =
db.Cinema.find({"acteurs":{"$elemMatch":{"nom":"Eastwood"}}}, {"nom":1, "_id":0})
```

```
while ( recordset.hasNext() ) {
```

```
    film = recordset.next() ;
```

```
    print ("- ", film.nom);
```

```
}
```



```

29
30 ## Question 10
31 ...
32 conn = new Mongo();
33
34 db = conn.getDB("db1");
35
36 recordset = db.Cinema.find({"acteurs":{"$elemMatch":{"nom":"Eastwood"}}},{ "nom":1,"_id":0})
37
38 while ( recordset.hasNext() ) {
39
40     film = recordset.next() ;
41
42     print("- ", film.nom);
43
44 }
45 ...

```

## 11. Question 11

```

db.User.insert({ pseudonyme : "Hico", preferences :
[ { cinema_id : "5e9c8debc3129268082dcb17", etoiles :
2 },

    { cinema_id : "5e9c8debc3129268082dcb18",
etoiles : 3 }

],

},

)

db.User.insert( { pseudonyme : "Bilingo",
preferences : [ { cinema_id :
"5e9c8debc3129268082dcb16", etoiles : 3 },

    { cinema_id : "5e9c8debc3129268082dcb17",
etoiles : 1 }

]

})

db.User.insert( { pseudonyme : "Bafana",
preferences : [ { cinema_id :
"5e9c8debc3129268082dcb19", etoiles : 2 },

    { cinema_id : "5e9c8debc3129268082dcb1b",
etoiles : 2 }

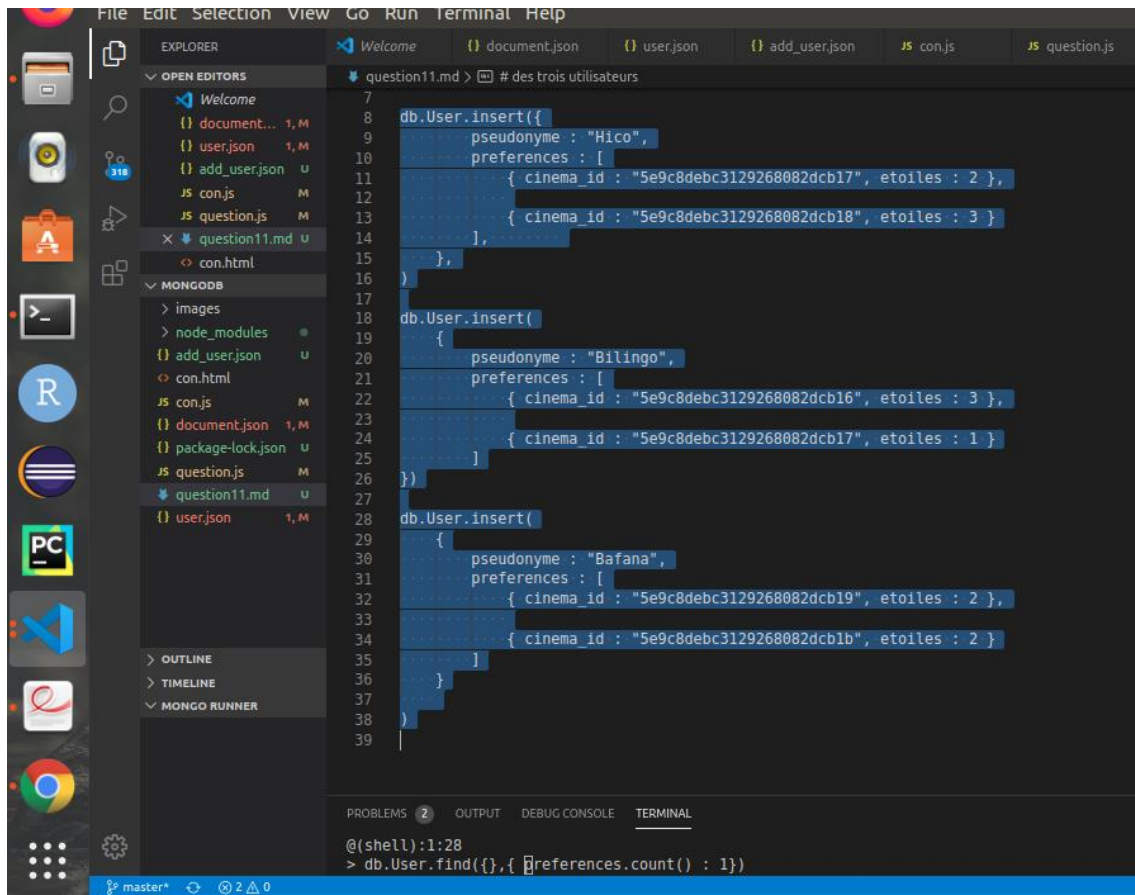
]

}

```



)



```
7
8 db.User.insert({
9   pseudonyme : "Hico",
10  preferences : [
11    { cinema_id : "5e9c8debc3129268082dcb17", etoiles : 2 },
12    { cinema_id : "5e9c8debc3129268082dcb18", etoiles : 3 }
13  ],
14 },
15 )
16
17 db.User.insert(
18 {
19   pseudonyme : "Bilingo",
20   preferences : [
21     { cinema_id : "5e9c8debc3129268082dcb16", etoiles : 3 },
22     { cinema_id : "5e9c8debc3129268082dcb17", etoiles : 1 }
23   ]
24 })
25
26 db.User.insert(
27 {
28   pseudonyme : "Bafana",
29   preferences : [
30     { cinema_id : "5e9c8debc3129268082dcb19", etoiles : 2 },
31     { cinema_id : "5e9c8debc3129268082dcb1b", etoiles : 2 }
32   ]
33 })
34
35
36
37
38
39
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

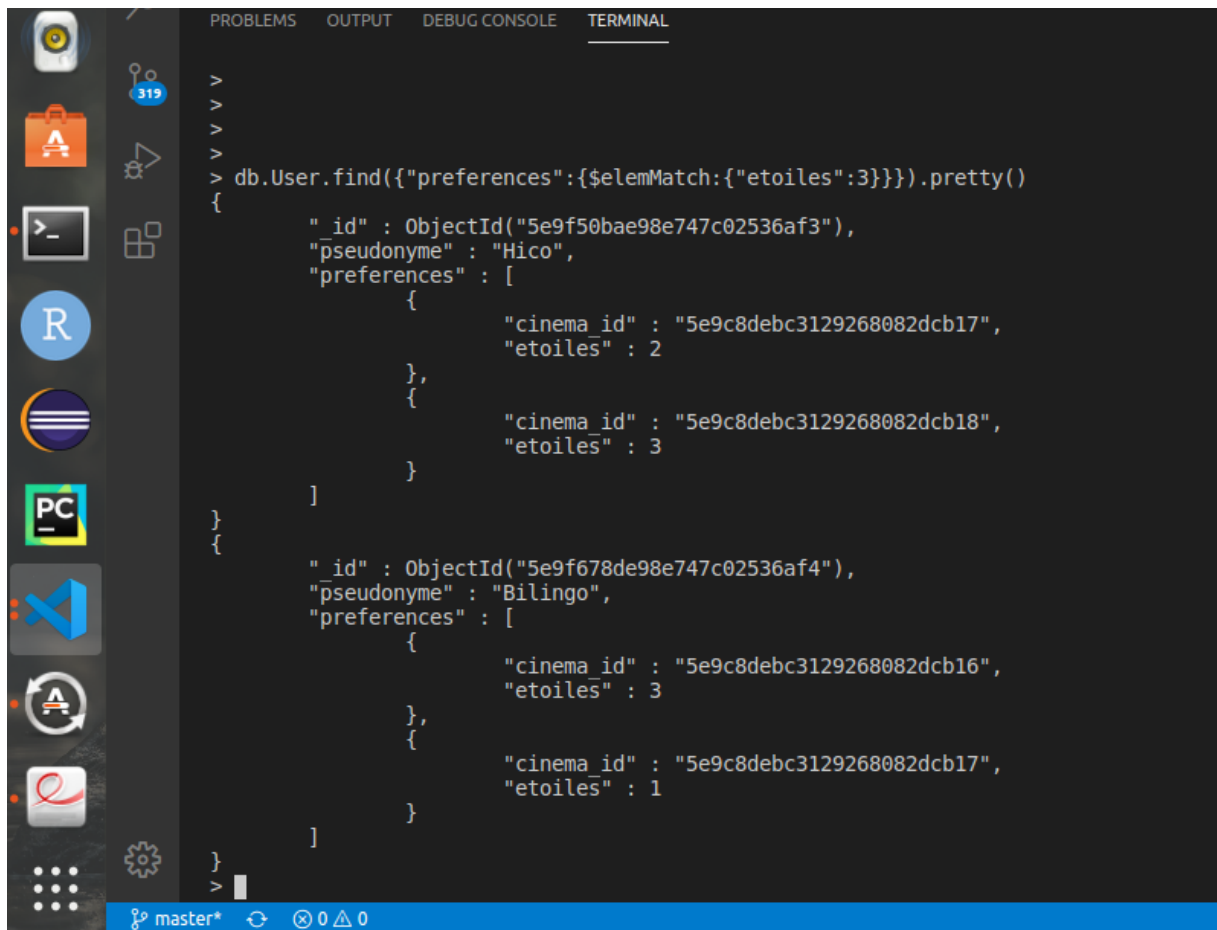
@(shell):1:28  
> db.User.find({}, { \$project: { references: 1 } })

## 12. Question 12

Les identifiants sont contrôlés par le système, ils sont générés au cours d'une insertion, mais s'il arrive qu'on change de base de données les identifiants seront différents.

## 13. Question 13

```
db.User.find({"preferences":{$elemMatch:{"etoiles":3}}})
```

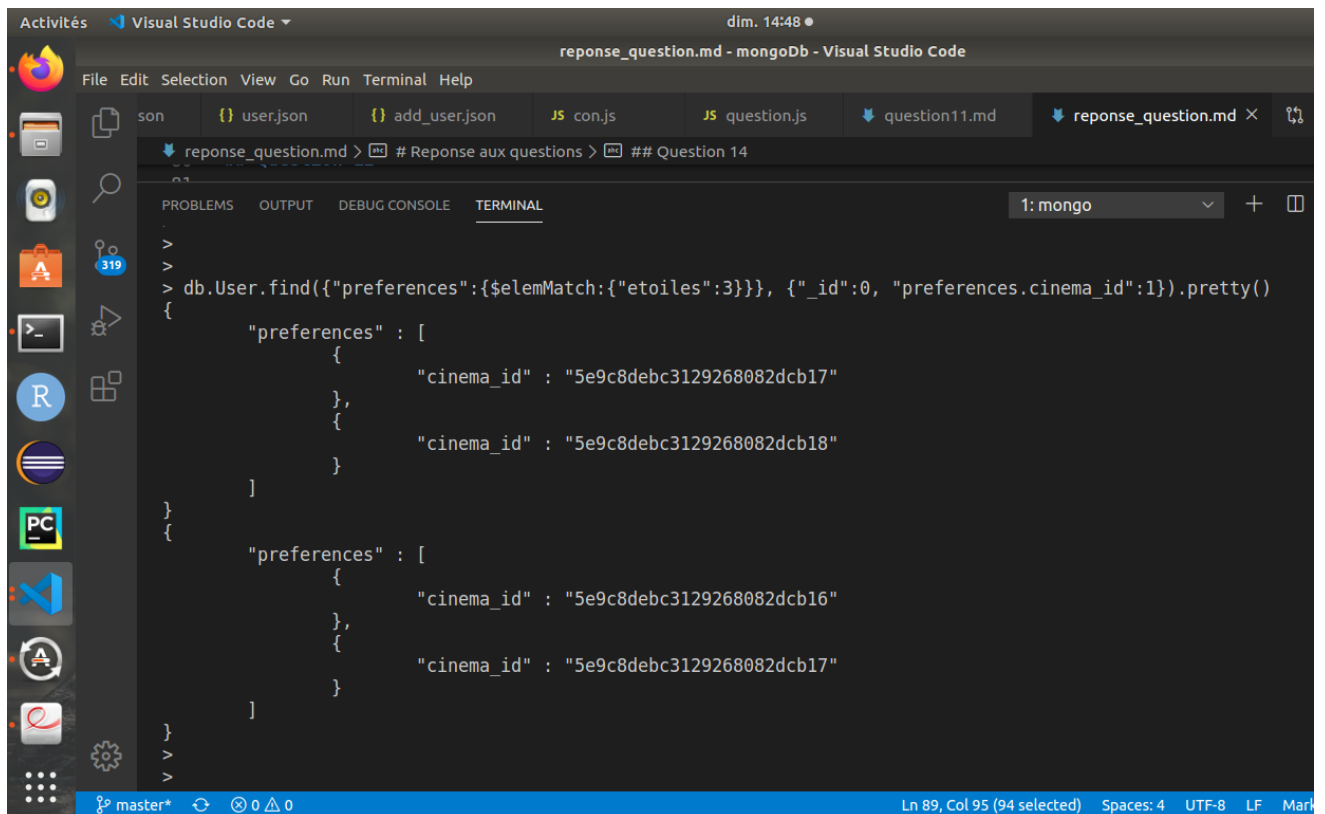


The screenshot shows a VS Code interface with a terminal window open. The terminal displays a MongoDB query and its results. The query is `db.User.find({"preferences":{"elemMatch":{"etoiles":3}}}).pretty()`. The results are two JSON objects representing users. The first user, "Hico", has two entries in their preferences list, both with 2 stars. The second user, "Bilingo", has two entries in their preferences list, with 3 stars and 1 star respectively. The status bar at the bottom indicates the current branch is "master".

```
>
>
>
>
> db.User.find({"preferences":{"elemMatch":{"etoiles":3}}}).pretty()
{
  "_id" : ObjectId("5e9f50bae98e747c02536af3"),
  "pseudonyme" : "Hico",
  "preferences" : [
    {
      "cinema_id" : "5e9c8debc3129268082dcb17",
      "etoiles" : 2
    },
    {
      "cinema_id" : "5e9c8debc3129268082dcb18",
      "etoiles" : 3
    }
  ]
},
{
  "_id" : ObjectId("5e9f678de98e747c02536af4"),
  "pseudonyme" : "Bilingo",
  "preferences" : [
    {
      "cinema_id" : "5e9c8debc3129268082dcb16",
      "etoiles" : 3
    },
    {
      "cinema_id" : "5e9c8debc3129268082dcb17",
      "etoiles" : 1
    }
  ]
}
>
```

#### 14. Question 14

`db.User.find({"preferences":{"elemMatch":{"etoiles":3}}}, {"_id":0, "preferences.cinema_id":1})`



```
reponse_question.md - mongoDb - Visual Studio Code
File Edit Selection View Go Run Terminal Help
son {} user.json {} add_user.json JS con.js JS question.js question11.md reponse_question.md
reponse_question.md > # Reponse aux questions > ## Question 14
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: mongo
>
> db.User.find({'preferences':{$elemMatch:{"etoiles":3}}}, {'_id':0, 'preferences.cinema_id':1}).pretty()
{
  "preferences" : [
    {
      "cinema_id" : "5e9c8debc3129268082dcb17"
    },
    {
      "cinema_id" : "5e9c8debc3129268082dcb18"
    }
  ]
},
{
  "preferences" : [
    {
      "cinema_id" : "5e9c8debc3129268082dcb16"
    },
    {
      "cinema_id" : "5e9c8debc3129268082dcb17"
    }
  ]
}
}
>
>
Ln 89, Col 95 (94 selected) Spaces: 4 UTF-8 LF Mark
```

## 15. Question 15

Comme Mongo ne gère pas de jointure, on va devoir la réaliser au niveau de la couche cliente applicative.

Interroger MongoDB avec la requête précédente

Récupérer et dédoubler la liste des identifiants de films

Boucler sur ces identifiants, créer une requête pour chaque identifiant et interroger MongoDB

## 16. Question 16

```
conn = new Mongo();
```

```
db = conn.getDB("mydb");
```

```
films = db.User.distinct("preferences.film", {"preferences":{$elemMatch:{"etoiles":3}}});
```

```
for (var i=0; i<films.length; i++) {
```

```
    var id = films[i];
    if (film = db.Cinema.findOne(ObjectId(id)))
        print(film.nom);
}
```

# TRAVAUX PRATIQUE I (NEO4j)

## 1. Question 1

```
CREATE(p:personnage { name : 'Gnanago', nickname : 'Brice' }) RETURN p
```



## 2. Question 2

```
CREATE(gregor:personnage { name : 'Gregor Clegane',  
nickname : 'The Mountain' })
```

```
CREATE(oberyn:personnage { name : 'Oberyn Martell',  
nickname : 'The Viper' })
```

```
CREATE (gregor)-[:TUE { type : 'duel' }]->(oberyn)  
RETURN gregor, oberyn
```



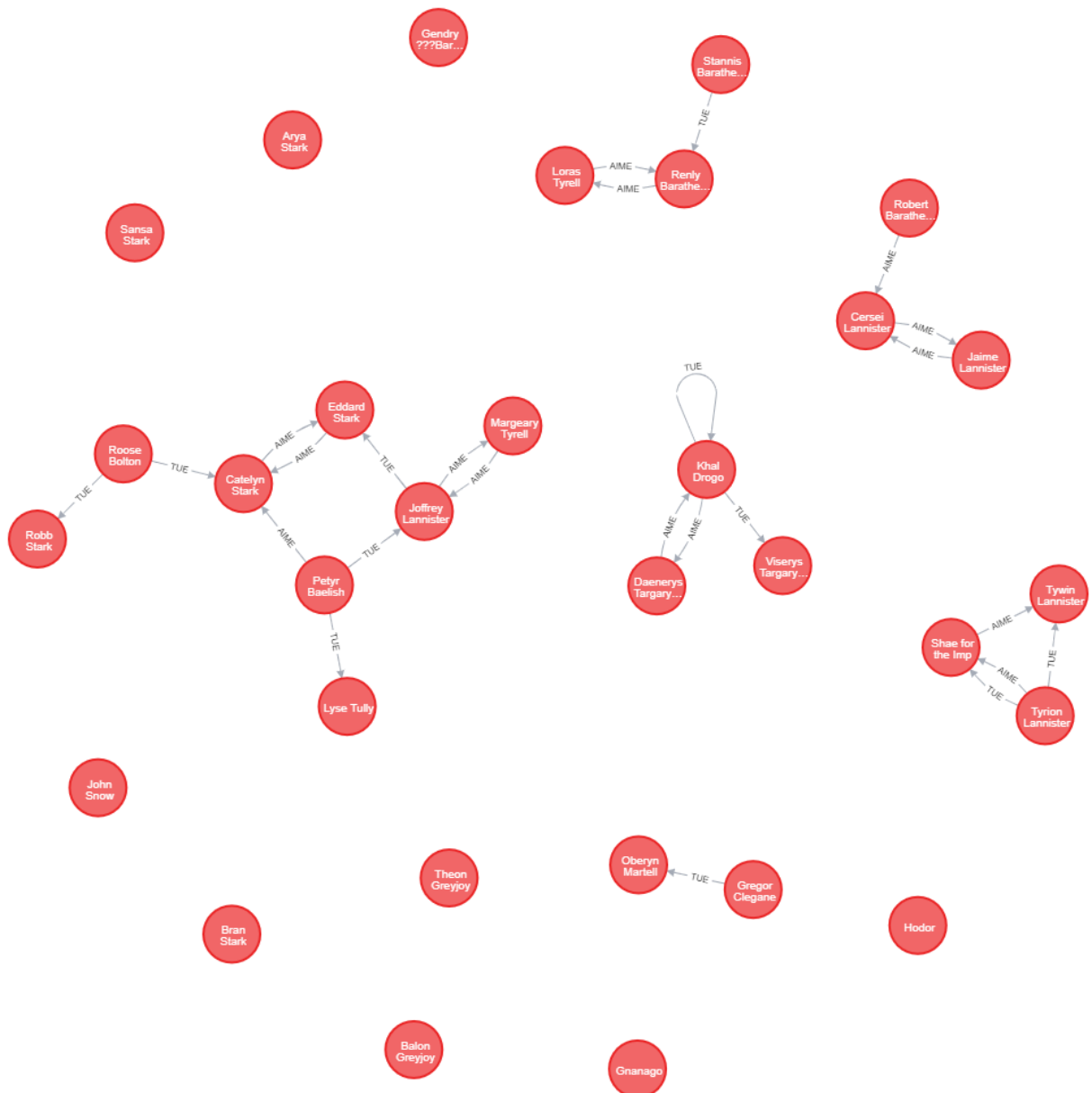
## 3. Question 3

```
match (p) return p
```



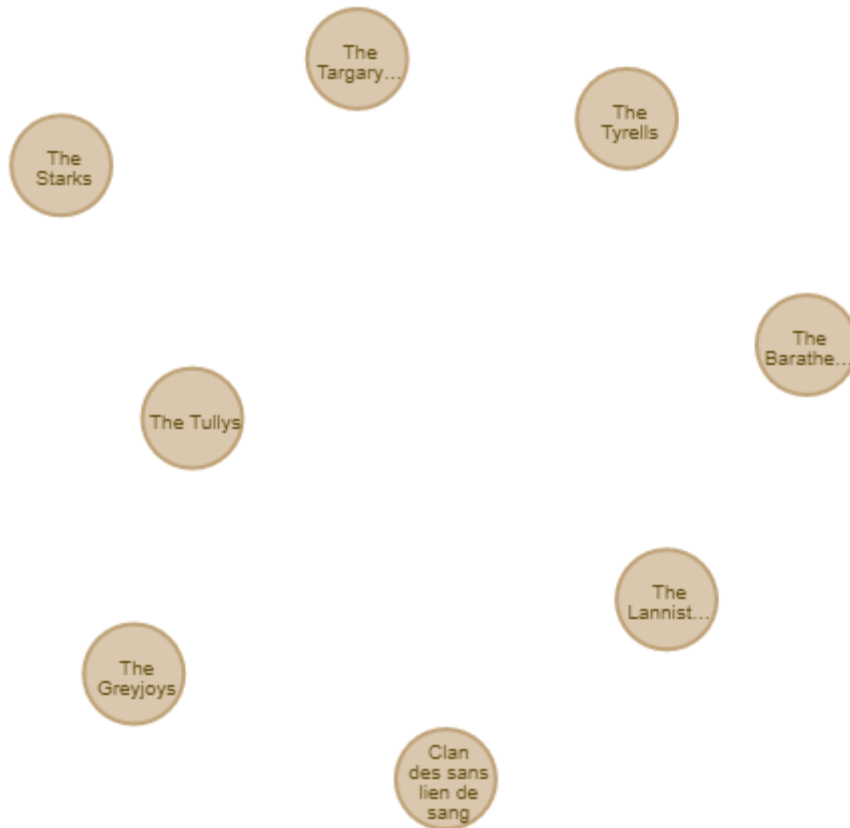
#### 4. Question 4

match (n) return n



## 5. Question 5

```
match (n:clan) return n
```



## 6. Question 6

```
match (n {nickname:'The Imp'}) return n
```



## 7. Question 7

```
match (c:clan {clanname:'The Starks'}) return c
```



### 8. Question 8

```
match (c:clan {clanname:'The Starks'})<-[1:LIER]-(p)
return l,c,p
```



## 9. Question 9

```
match (:personnage) -[res:TUE] -> (:personnage)
```

```
return res
```

