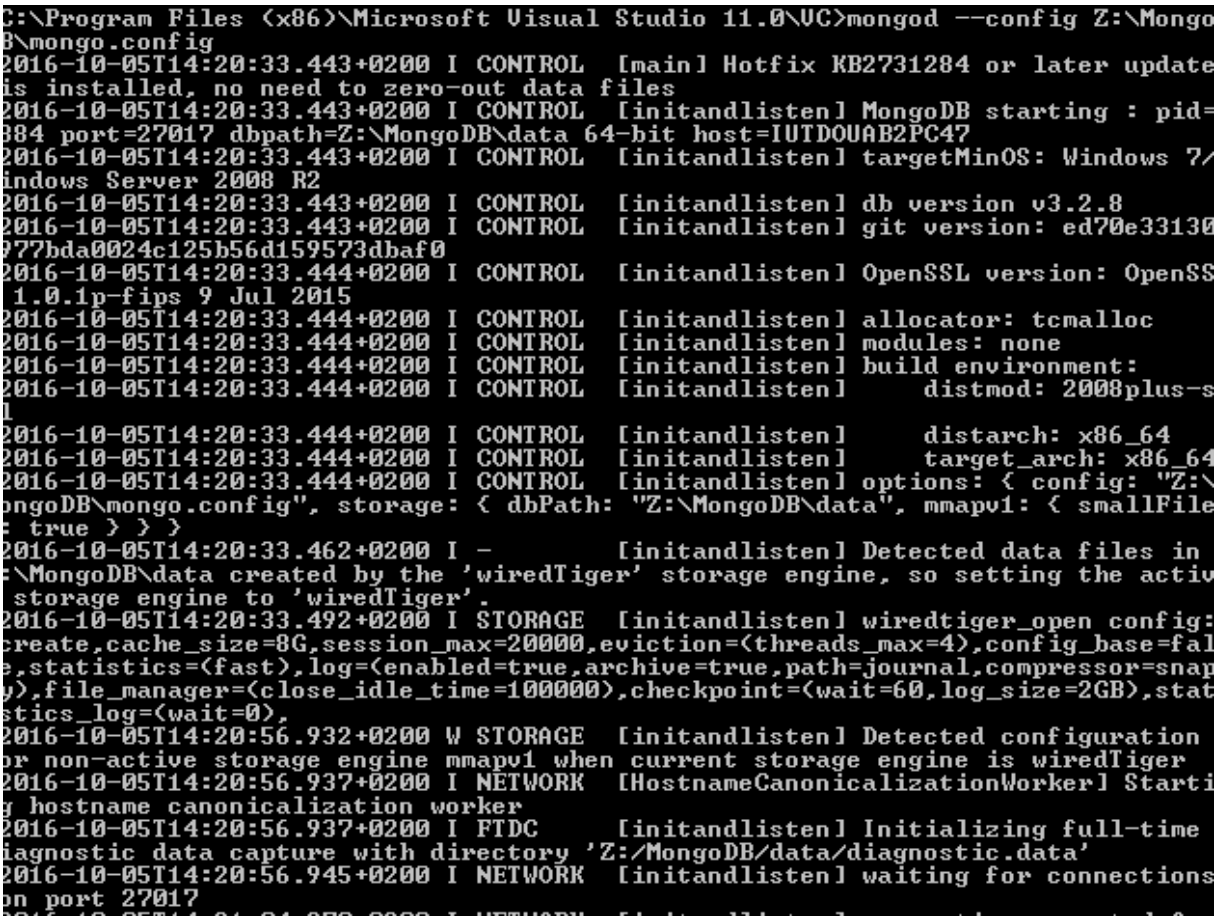
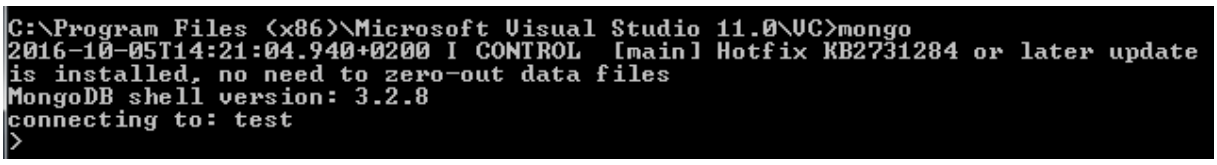
TP-MongoDB

(TOUMI & BAKAYOKO)

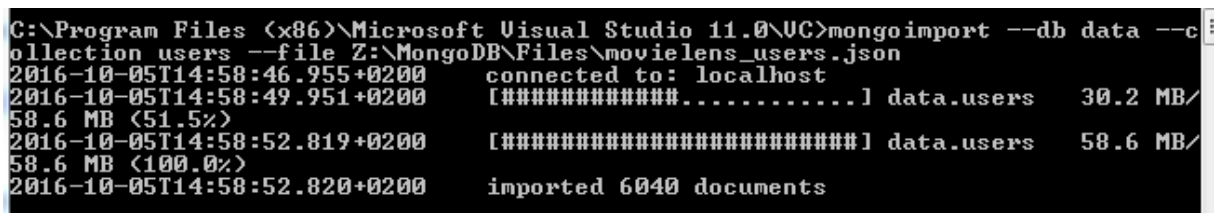
serveur



client



Import des collections dans ma database data



```
C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC>mongoimport --db data --c
ollection movies --file Z:\MongoDB\Files\movielens_movies.json
2016-10-05T15:05:20.636+0200    connected to: localhost
2016-10-05T15:05:20.904+0200    imported 3883 documents
```

Question 1

```
> use data
switched to db data
> db.users.count()
6040
```

On voit bien qu'il y a autant de users que de documents contenue dans la collection users importés soit 6040.

Question 2

```
> db.movies.count()
3883
```

On voit bien qu'il y a autant de movies que de documents contenue dans la collection movies importés soit 3883.

Question 3

```
> db.users.find({name:"Clifford Johnathan"},{name:1,occupation:1,_id:0})
{ "name" : "Clifford Johnathan", "occupation" : "technician/engineer" }
```

Question 4

```
> db.users.find({age:{$gt:17, $lt:31}}).count()
2365
```

Question 5

```
> db.users.find({occupation:{$in:["artist","scientist"]}}).count()
411
```

Question 6

```
> db.users.find({gender:"F"},{name:1, gender:1, _id:0, age:1}).sort({age:-1}).li
mit(10)
{ "name" : "Jestine Booker", "gender" : "F", "age" : 99 }
{ "name" : "Babara Elden", "gender" : "F", "age" : 98 }
{ "name" : "Susanna Shaun", "gender" : "F", "age" : 96 }
{ "name" : "Yaeko Hassan", "gender" : "F", "age" : 95 }
{ "name" : "Linh Tyrell", "gender" : "F", "age" : 95 }
{ "name" : "Ka Joe", "gender" : "F", "age" : 94 }
{ "name" : "Lashandra Sal", "gender" : "F", "age" : 94 }
{ "name" : "Starla Desmond", "gender" : "F", "age" : 94 }
{ "name" : "Lakeisha Wilbur", "gender" : "F", "age" : 94 }
{ "name" : "Aleshia Carol", "gender" : "F", "age" : 94 }
>
```

Question 7

```
> db.users.distinct("occupation")
[
        "other",
        "academic/educator",
        "doctor/health care",
        "scientist",
        "sales/marketing",
        "college/grad student",
        "retired",
        "executive/managerial",
        "technician/engineer",
        "self-employed",
        "programmer",
        "homemaker",
        "writer",
        "customer service",
        "clerical/admi",
        "K-12 student",
        "lawyer",
        "artist",
        "unemployed",
        "tradesman/craftsman",
        "farmer"
]
>
```

Question 8

```
> db.users.insert({name:"Nacereddine",gender:"M", age:21,occupation:"lawyer"})
WriteResult({ "nInserted" : 1 })
> db.users.find({name:"Nacereddine"})
{ "_id" : ObjectId("57fcfbe60f0ee58bc70805dd"), "name" : "Nacereddine", "gender"
 : "M", "age" : 21, "occupation" : "lawyer" }
```

Question 9

```
> db.users.update({name:"Nacereddine"},{$addToSet:{movies:[{movieid:145, rating:
5}]}},{multi : true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.users.find({name:"Nacereddine"})
{ "_id" : ObjectId("57fcfbe60f0ee58bc70805dd"), "name" : "Nacereddine", "gender"
 : "M", "age" : 21, "occupation" : "lawyer", "movies" : [ [ { "movieid" : 145, "
rating" : 5 } ] ] }
```

Question 10

```
> db.users.remove({name:"Nacereddine"})
WriteResult({ "nRemoved" : 1 })
```

Question 11

```
> db.users.update({occupation:"programmer"},{$set:{occupation:"developer"}},{mul
ti : true})
WriteResult({ "nMatched" : 388, "nUpserted" : 0, "nModified" : 388 })
```

Question 12

```
> db.movies.find({title:{$regex:/198.*/}}).count()
598
```

Question 13

```
> db.movies.find({$or:[{title:{$regex:/198[4-9]/}},{title:{$regex:/199[0-2]/}}]}
).count()
668
```

Question 14

```
> db.movies.find({genres:{$regex:"Horror"}}).count()
343
>
```

Question 15

```
> db.movies.find({$and:[{genres:{$regex:"Musical"}},{genres:{$regex:"Romance"}}]
}).count()
18
```

Question 16

```
> db.movies.find().forEach( function(ch){ch.year = ch.title.substr(ch.title.leng
th-5,4);db.movies.save(ch);})
> db.movies.find().forEach( function(ch){ch.title = ch.title.substr(0,ch.title.l
ength-7);db.movies.save(ch);})
> db.movies.find
```

Verification :

```
> db.movies.find()
{ "_id" : 2, "title" : "Jumanji", "genres" : "Adventure|Children's|Fantasy", "y
ar" : "1995" }
```

Question 17

```
> db.movies.find().forEach( function(ch){ch.genres = ch.genres.split('|');db.mov
ies.save(ch);})
> db.movies.find()
{ "_id" : 2, "title" : "Jumanji", "genres" : [ "Adventure", "Children's", "Fanta
sy" ], "year" : "1995" }
```

Question 18

```
> db.users.find().forEach(function(ch){ch.movies.forEach(function(mov){mov.date
= new Date(mov.timestamp*1000); delete mov.timestamp;});db.users.save(ch);})
```

Question 19

```
> db.users.find({movies:{$elemMatch:{movieid:1196}}}).count()
2990
```

Question 20

```
> db.users.find({"movies.movieid":{$all:[260,1196,1210]}}).count()
1926
```

Question 21

```
> db.users.find({movies:{$size:48}}).count()
51
```

Question 22

```
> db.users.find().forEach(function(ch){ch.num_ratings=ch.length;db.users.save(ch
);})
```

```
> db.users.find().forEach(function(ch){ch.num_ratings=ch.movies.length;db.users.
save(ch);})
```

Question 23

```
> db.users.find({num_ratings:{$gt:90}}).count()
3114
```

Question 24

```
> db.users.find({movies:{$elemMatch:{date:{$gte:ISODate("2001-01-01T00:00:00Z")}
}}}).count()
1177
```

Question 25

```
> db.users.find({name:"Jayson Brad"},{movies:{$slice:[{movies:{$size:1}}-5,5]}})

{ "_id" : 6016, "name" : "Jayson Brad", "gender" : "M", "age" : 47, "occupation"
 : "academic/educator", "movies" : [ { "movieid" : 2053, "rating" : 1, "date" :
ISODate("2000-04-26T20:04:01Z") }, { "movieid" : 2054, "rating" : 3, "date" : IS
ODate("2000-04-26T19:48:30Z") }, { "movieid" : 3795, "rating" : 3, "date" : ISOD
ate("2001-07-06T21:14:25Z") }, { "movieid" : 2059, "rating" : 3, "date" : ISODat
e("2000-04-26T20:02:09Z") }, { "movieid" : 580, "rating" : 4, "date" : ISODate("
2000-04-28T22:04:58Z") } ], "num_ratings" : 909 }
>
```

Question 26

```
> db.users.find({name:"Tracy Edward"},{movies:{$elemMatch:{movieid:1210}}})
```

```
"rating" : 5
```

Question 27

```
> db.users.find({movies:{$elemMatch:{movieid:2194, rating:5}}}).count()
317
```

Question 28

```
> db.users.update({name:"Barry Erin"},{$inc:{num_ratings:1},$push:{movies:{movie
id:14, rating:4,date:ISODate("2016-10-03T18:45:01Z")}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.users.find({name:"Barry Erin"},{movies:{$elemMatch:{movieid:14}}})
{ "_id" : 6040, "movies" : [ { "movieid" : 14, "rating" : 4, "date" : ISODate("2
016-10-03T18:45:01Z") } ] }
```

Question 29

```
> db.users.find({name:"Marquis Billie"},{movies:{$elemMatch:{movieid:1311}}})
{ "_id" : 58, "movies" : [ { "movieid" : 1311, "rating" : 1, "date" : ISODate("2
000-12-27T17:02:08Z") } ] }
```

```
> db.users.update({name:"Marquis Billie"},{$pull:{movies:{movieid:1311}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

On voit bien qu'il a été supprimé car il nous renvoie seulement l'id de Billie

```
> db.users.find({name:"Marquis Billie"},{movies:{$elemMatch:{movieid:1311}}})
{ "_id" : 58 }
```

Question 30

```
> db.movies.find({title:"Cinderella"})
{ "_id" : 1022, "title" : "Cinderella", "genres" : [ "Children's", "Musical" ],
"year" : "1950" }
>
```

```
> db.movies.update({title:"Cinderella"},{$set:{genres:["Animation","Children's",
"Musical"]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movies.find({title:"Cinderella"})
{ "_id" : 1022, "title" : "Cinderella", "genres" : [ "Animation", "Children's",
"Musical" ], "year" : "1950" }
```

Question 31

```
> db.users.find().snapshot().forEach(function(colusers){colusers.movies.forEach(
function(movies){movies.movieref={"$ref":"movies","$id":movies.movieid};});db.us
ers.update({_id:colusers._id},{$set:{movies:colusers.movies}})})
```

On vérifie :

```
> db.users.findOne()
```

```
{
        "movieid" : 1237,
        "rating" : 5,
        "date" : ISODate("2000-04-25T23:48:25Z"),
        "movieref" : DBRef("movies", 1237)
},
```

Question 32

```
> var ratedTaxi =db.movies.findOne({title:"Taxi Driver"})._id;db.users.count({mo
vies:{$elemMatch:{"movieref.$id":ratedTaxi}}})
1240
```

Question 33

```
> var ratedTaxi =db.movies.findOne({title:"Taxi Driver"})._id;db.users.count({mo
vies:{$elemMatch:{"movieref.$id":ratedTaxi, "rating":5}}})
538
```

Question 34

```
> db.users.find({gender:"F"},{_id:0,name:1,gender:1}).sort({"movies.date":1}).li
mit(10)
{ "name" : "Raquel Stanton", "gender" : "F" }
{ "name" : "Yaeko Hassan", "gender" : "F" }
{ "name" : "Anastasia Norbert", "gender" : "F" }
{ "name" : "Demetrice Bert", "gender" : "F" }
{ "name" : "Ileen Francis", "gender" : "F" }
{ "name" : "Anissa Jeffery", "gender" : "F" }
{ "name" : "Lourie Ira", "gender" : "F" }
{ "name" : "Jana Jame", "gender" : "F" }
{ "name" : "Ronni Frank", "gender" : "F" }
{ "name" : "Sandie Alan", "gender" : "F" }
```

Avec le explain() :

```
> db.users.find({gender:"F"},{_id:0,name:1,gender:1}).sort({"movies.date":1}).li
mit(10).explain()
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "data.users",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "gender" : {
                                "$eq" : "F"
                        }
                },
                "winningPlan" : {
                        "stage" : "PROJECTION",
                        "transformBy" : {
                                "_id" : 0,
                                "name" : 1,
                                "gender" : 1
                        },
                        "inputStage" : {
                                "stage" : "SORT",
                                "sortPattern" : {
                                        "movies.date" : 1
                                },
                                "limitAmount" : 10,
                                "inputStage" : {
                                        "stage" : "SORT_KEY_GENERATOR",
                                        "inputStage" : {
                                                "stage" : "COLLSCAN",
                                                "filter" : {
                                                        "gender" : {
                                                                "$eq" : "F"
                                                        }
                                                },
                                                "direction" : "forward"
                                        }
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "IUTDOUAB2PC47",
                "port" : 27017,
                "version" : "3.2.8",
                "gitVersion" : "ed70e33130c977bda0024c125b56d159573dbaf0"
        },
        "ok" : 1
}
```

Question 35

Requete et verifaction :

```
> db.collection.createIndex({gender:2, "movies.date":3})
{
        "createdCollectionAutomatically" : true,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.users.getIndexes()
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "data.users"
        }
]
```

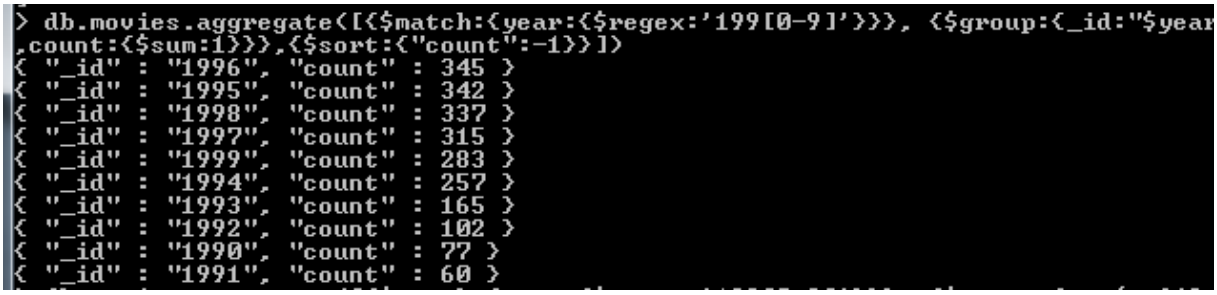
Question 36

Réexécution de la question 34 avec le explain() :

```
> db.users.find({gender:"F"},{_id:0,name:1,gender:1}).sort({"movies.date":1}).li
mit(10).explain()
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "data.users",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "gender" : {
                                "$eq" : "F"
                        }
                },
                "winningPlan" : {
                        "stage" : "PROJECTION",
                        "transformBy" : {
                                "_id" : 0,
                                "name" : 1,
                                "gender" : 1
                        },
                        "inputStage" : {
                                "stage" : "SORT",
                                "sortPattern" : {
                                        "movies.date" : 1
                                },
                                "limitAmount" : 10,
                                "inputStage" : {
                                        "stage" : "SORT_KEY_GENERATOR",
                                        "inputStage" : {
                                                "stage" : "COLLSCAN",
                                                "filter" : {
                                                        "gender" : {
                                                                "$eq" : "F"
                                                        }
                                                },
                                                "direction" : "forward"
                                        }
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "IUTDOUAB2PC47",
                "port" : 27017,
                "version" : "3.2.8",
                "gitVersion" : "ed70e33130c977bda0024c125b56d159573dbaf0"
        },
        "ok" : 1
}
```

On peut voir qu'il n'y a pas d'indexes.

Question 37



```
> db.movies.aggregate([{$match:{year:{$regex:'199[0-9]'}}}, {$group:{_id:"$year
,count:{$sum:1}}},{$sort:{"count":-1}}])
{ "_id" : "1996", "count" : 345 }
{ "_id" : "1995", "count" : 342 }
{ "_id" : "1998", "count" : 337 }
{ "_id" : "1997", "count" : 315 }
{ "_id" : "1999", "count" : 283 }
{ "_id" : "1994", "count" : 257 }
{ "_id" : "1993", "count" : 165 }
{ "_id" : "1992", "count" : 102 }
{ "_id" : "1990", "count" : 77 }
{ "_id" : "1991", "count" : 60 }
```

Question 38

Requête :

```
> db.users.aggregate([{$project:{movies:{$filter:{input:"$movies",as:"mv", cond:
{$eq:["$$mv.movieid",296]}}}}},{$project:{movies:1,FilmNb:{$size:"$movies"}}},{$
match:{FilmNb:1}},{$project:{movies:1}},{$project:{rateOfMovie:{$arrayElemAt:["$
movies",0]}}},{$project:{FinalRateOfMovie:"$rateOfMovie.rating"}},{$group:{_id:2
96,rateMovieAvg:{$avg:"$FinalRateOfMovie"}}}])
{ "_id" : 296, "rateMovieAvg" : 4.278212805158913 }
```

Resultat :

```
{ "_id" : 296, "rateMovieAvg" : 4.278212805158913 }
```

Question 39

```
> db.users.aggregate([{$project:{name:1,MaxRate:{$max:"movies.rating"},MinRate:{
$min:"$movies.rating"},AvgRating:{$avg:"$movies.rating"}}},{$sort:{AvgRating:1}}
])
```

Resultat :

```
{ "_id" : 3598, "name" : "Billie Evan", "MaxRate" : "movies.rating", "MinRate" :
 1, "AvgRating" : 1.0153846153846153 }
{ "_id" : 4486, "name" : "Logan Kendrick", "MaxRate" : "movies.rating", "MinRate
" : 1, "AvgRating" : 1.0588235294117647 }
{ "_id" : 2744, "name" : "Ty Hank", "MaxRate" : "movies.rating", "MinRate" : 1,
"AvgRating" : 1.3043478260869565 }
{ "_id" : 4539, "name" : "Loyd Winfred", "MaxRate" : "movies.rating", "MinRate"
: 1, "AvgRating" : 1.815126050420168 }
{ "_id" : 5850, "name" : "Noelia Cordell", "MaxRate" : "movies.rating", "MinRate
" : 1, "AvgRating" : 1.8448275862068966 }
{ "_id" : 5334, "name" : "Vivien Al", "MaxRate" : "movies.rating", "MinRate" : 1
, "AvgRating" : 1.9272727272727272 }
{ "_id" : 4349, "name" : "Rodrigo Gerald", "MaxRate" : "movies.rating", "MinRate
" : 1, "AvgRating" : 1.962962962962963 }
{ "_id" : 4636, "name" : "Jeramy Wyatt", "MaxRate" : "movies.rating", "MinRate"
: 1, "AvgRating" : 2 }
{ "_id" : 5686, "name" : "Cristopher Everett", "MaxRate" : "movies.rating", "Min
Rate" : 1, "AvgRating" : 2.0452830188679245 }
{ "_id" : 3209, "name" : "Jeromy Levi", "MaxRate" : "movies.rating", "MinRate" :
 1, "AvgRating" : 2.0608695652173914 }
{ "_id" : 1608, "name" : "Romeo Jeff", "MaxRate" : "movies.rating", "MinRate" :
1, "AvgRating" : 2.0833333333333335 }
{ "_id" : 4575, "name" : "Celinda Porfirio", "MaxRate" : "movies.rating", "MinRa
te" : 1, "AvgRating" : 2.088 }
{ "_id" : 4916, "name" : "Logan Levi", "MaxRate" : "movies.rating", "MinRate" :
1, "AvgRating" : 2.088235294117647 }
{ "_id" : 1747, "name" : "Stacey Levi", "MaxRate" : "movies.rating", "MinRate" :
 1, "AvgRating" : 2.138888888888889 }
{ "_id" : 1761, "name" : "Houston Willard", "MaxRate" : "movies.rating", "MinRat
e" : 1, "AvgRating" : 2.15929203539823 }
{ "_id" : 1340, "name" : "Jake Lonny", "MaxRate" : "movies.rating", "MinRate" :
1, "AvgRating" : 2.1627329192546583 }
{ "_id" : 163, "name" : "Joe Marcus", "MaxRate" : "movies.rating", "MinRate" : 1
, "AvgRating" : 2.1828793774319064 }
{ "_id" : 1100, "name" : "Travis Bryon", "MaxRate" : "movies.rating", "MinRate"
: 1, "AvgRating" : 2.1988188976377954 }
{ "_id" : 5944, "name" : "Tomoko Barrett", "MaxRate" : "movies.rating", "MinRate
" : 1, "AvgRating" : 2.2 }
{ "_id" : 5039, "name" : "Audie Miquel", "MaxRate" : "movies.rating", "MinRate"
: 1, "AvgRating" : 2.202777777777778 }
Type "it" for more
```