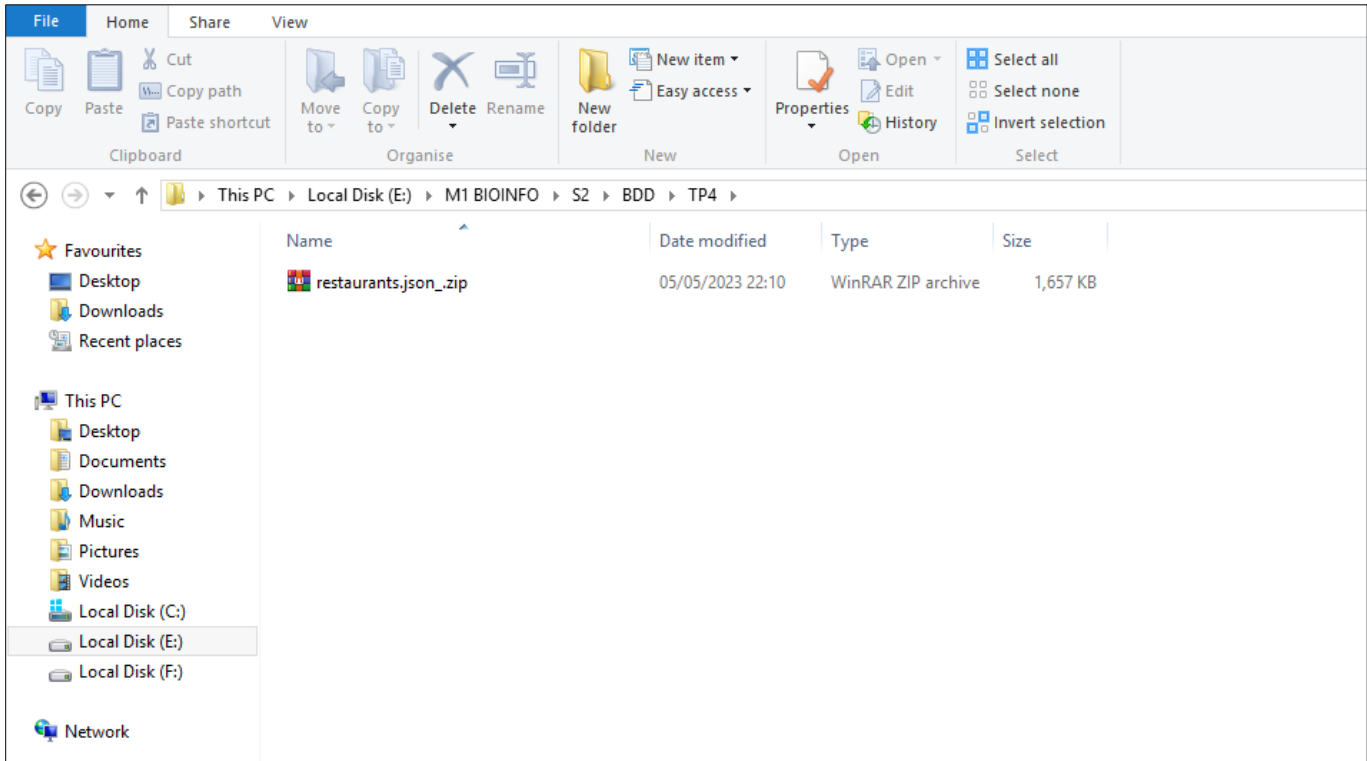


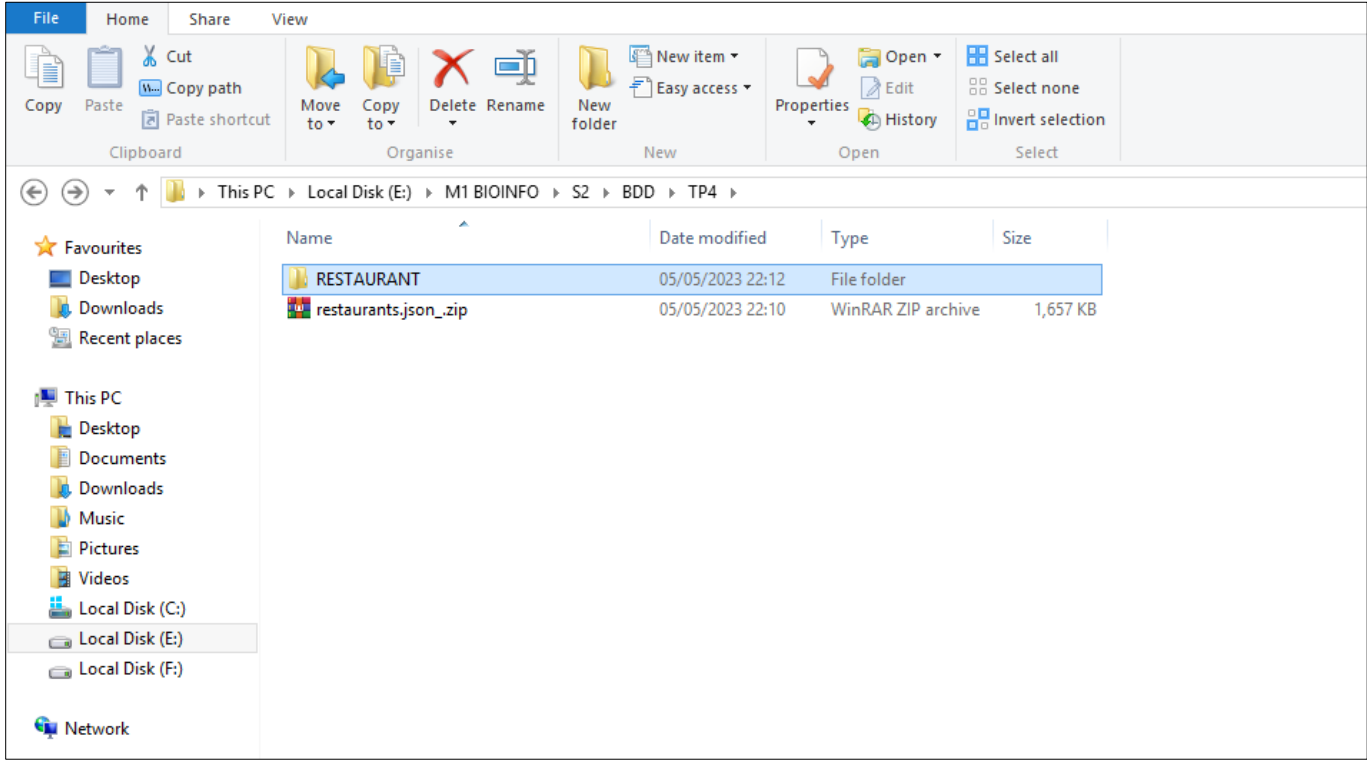
TD/TP4 : TP MongoDB

PARTIE 1 : CONTRAINTES D'INTEGRITE STATIQUES

1. Téléchargez l'archive suivante : restaurants.zip

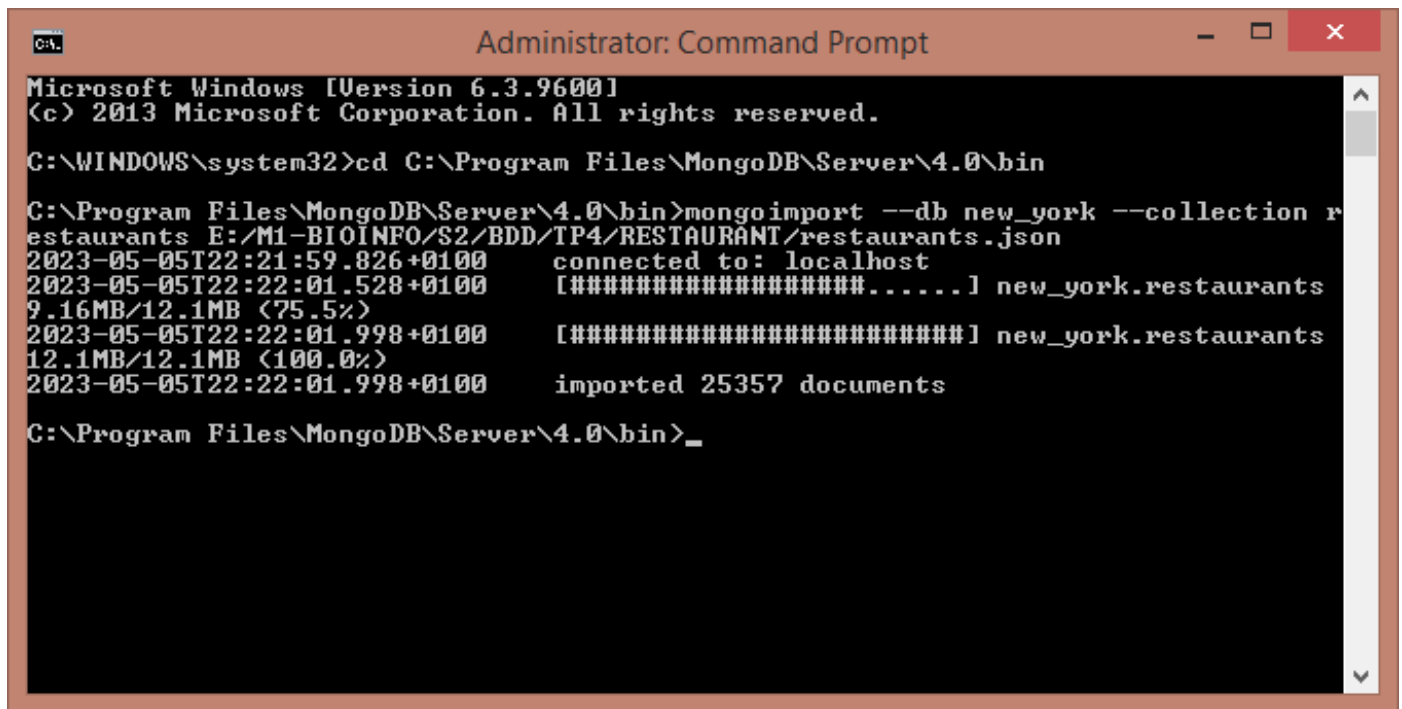


2. Décompresser l'archive (nommé le répertoire utilisé \$RESTAURANT)



3. Nous allons créer une base de données "new_york" (paramètre --db) et une collection "restaurants" (paramètre --collection). Attention, il ne faut pas de majuscules !
4. Dans une console (Windows : invite de commande, Linux : Shell/Konsole), aller dans le répertoire \$MONGO/bin
5. Exécutez la commande suivante :

```
mongoimport --db new_york --collection restaurants E:/M1-BIOINFO/S2/BDD/TP4/RESTAURANT/restaurants.json
```



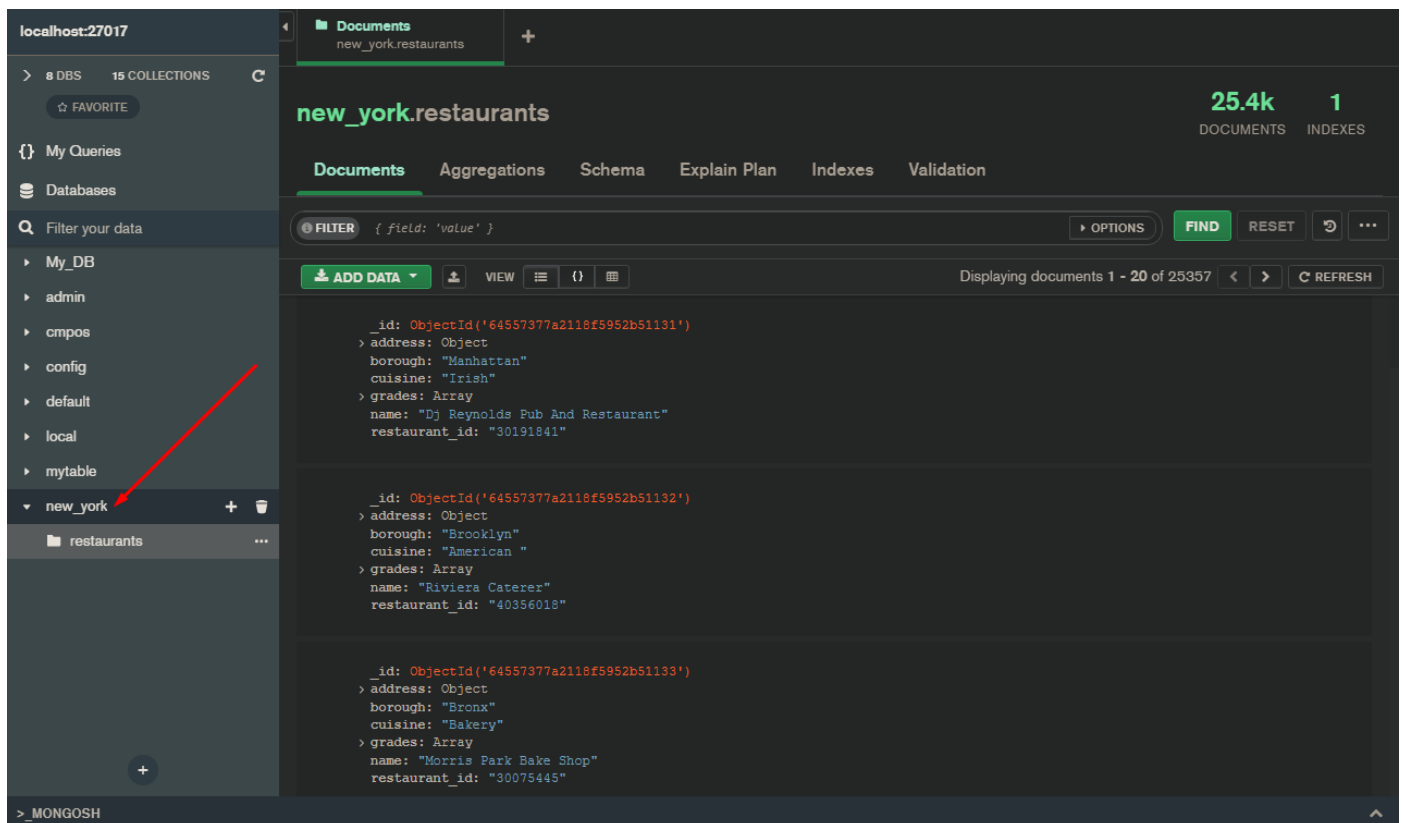
```
Administrator: Command Prompt

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Program Files\MongoDB\Server\4.0\bin

C:\Program Files\MongoDB\Server\4.0\bin>mongoimport --db new_york --collection r
estaurants E:/M1-BIOINFO/S2/BDD/TP4/RESTAURANT/restaurants.json
2023-05-05T22:21:59.826+0100    connected to: localhost
2023-05-05T22:22:01.528+0100    [#####] new_york.restaurants
9.16MB/12.1MB (75.5%)
2023-05-05T22:22:01.998+0100    [#####] new_york.restaurants
12.1MB/12.1MB (100.0%)
2023-05-05T22:22:01.998+0100    imported 25357 documents

C:\Program Files\MongoDB\Server\4.0\bin>_
```



6. Afficher un élément au hasard de la collection (findOne())

```
> _MONGOSH
> use new_york
< 'switched to db new_york'
> db.restaurants.findOne()
< { _id: ObjectId("64557377a2118f5952b51131"),
  address:
    { building: '351',
      coord:
        { type: 'Point',
          coordinates: [ -73.98513559999999, 40.7676919 ] },
      street: 'West 57 Street',
      zipcode: '10019' },
    borough: 'Manhattan',
    cuisine: 'Irish',
    grades:
      [ { date: 2014-09-06T00:00:00.000Z, grade: 'A', score: 2 },
        { date: 2013-07-22T00:00:00.000Z, grade: 'A', score: 11 },
        { date: 2012-07-31T00:00:00.000Z, grade: 'A', score: 12 },
        { date: 2011-12-29T00:00:00.000Z, grade: 'A', score: 12 } ],
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841' }
```

7. Afficher les noms de restaurants du quartier (borough) de Brooklyn.

```
> _MONGOSH
> db.restaurants.find({ "borough": "Brooklyn" })
< { _id: ObjectId("64557377a2118f5952b51132"),
  address:
    { building: '2780',
      coord:
        { type: 'Point',
          coordinates: [ -73.98241999999999, 40.579505 ] },
      street: 'Stillwell Avenue',
      zipcode: '11224' },
    borough: 'Brooklyn',
    cuisine: 'American ',
    grades:
      [ { date: 2014-06-10T00:00:00.000Z, grade: 'A', score: 5 },
        { date: 2013-06-05T00:00:00.000Z, grade: 'A', score: 7 },
        { date: 2012-04-13T00:00:00.000Z, grade: 'A', score: 12 },
        { date: 2011-10-12T00:00:00.000Z, grade: 'A', score: 12 } ],
    name: 'Riviera Caterer',
    restaurant_id: '40356018' }
{ _id: ObjectId("64557377a2118f5952b51137"),
  address:
    { building: '6409',
```

- Donner le nombre de résultat (count)

```
> _MONGOSH
Type "it" for more
> db.restaurants.find({ "borough" : "Brooklyn" }).count()
< 6085
```

- Puis ajouter le critère : cuisine : Italian.

```
> db.restaurants.find({ "borough" : "Brooklyn",
  "cuisine" : "Italian" })
< { _id: ObjectId("64557377a2118f5952b51166"),
  address:
    { building: '10004',
      coord:
        { type: 'Point',
          coordinates: [ -74.03400479999999, 40.6127077 ] },
      street: '4 Avenue',
      zipcode: '11209' },
    borough: 'Brooklyn',
    cuisine: 'Italian',
    grades:
      [ { date: 2014-02-25T00:00:00.000Z, grade: 'A', score: 12 },
        { date: 2013-06-27T00:00:00.000Z, grade: 'A', score: 7 },
        { date: 2012-12-03T00:00:00.000Z, grade: 'A', score: 10 },
        { date: 2011-11-09T00:00:00.000Z, grade: 'A', score: 12 } ],
    name: 'Philadelphia Grille Express',
    restaurant_id: '40364305' }
  { _id: ObjectId("64557378a2118f5952b511a7"),
```

- Puis ajouter le critère de l'avenue (street) (attribut composant l'adresse)

```
> _MONGOSH
Type "it" for more
> db.restaurants.find({
  "borough" : "Brooklyn",
  "cuisine" : "Italian",
  "address.street" : "Quentin Road"
})
< { _id: ObjectId("64557378a2118f5952b514b3"),
  address:
    { building: '4220',
      coord: { type: 'Point', coordinates: [ -73.931917, 40.6175723 ] },
      street: 'Quentin Road',
      zipcode: '11234' },
    borough: 'Brooklyn',
    cuisine: 'Italian',
    grades:
      [ { date: 2014-07-10T00:00:00.000Z, grade: 'A', score: 9 },
        { date: 2013-04-25T00:00:00.000Z, grade: 'A', score: 11 },
        { date: 2012-03-14T00:00:00.000Z, grade: 'A', score: 10 } ],
    name: 'Salvi Restaurant',
    restaurant_id: '40394028' }
new_york>
```

8. A partir de la requête précédente :
- projeter le nom (name)

```
> _MONGOSH

> db.restaurants.find({
  "borough" : "Brooklyn",
  "cuisine" : "Italian",
  "address.street" : "Quentin Road"
},
  {"name":1}
)
< { _id: ObjectId("64557378a2118f5952b514b3"),
  name: 'Salvi Restaurant' }
new_york>
```

- enlever la projection par défaut du Id

```
> _MONGOSH

> db.restaurants.find({
  "borough" : "Brooklyn",
  "cuisine" : "Italian",
  "address.street" : "Quentin Road"
},
  {"name":1, "_id":0}
)
< { name: 'Salvi Restaurant' }
new_york>
```

- projeter les scores d'hygiène attribués par les commission d'inspection: grades.score.

```
> _MONGOSH

> db.restaurants.find({
  "borough" : "Brooklyn",
  "cuisine" : "Italian",
  "address.street" : "Quentin Road"
},
  {"name":1, "_id":0, "grades.score":1}
)
< { grades: [ { score: 9 }, { score: 11 }, { score: 10 } ],
  name: 'Salvi Restaurant' }
new_york>
```

9. Donner les noms et scores des restaurants de Manhattan ayant un score inférieur à 10.

```
>_MONGOSH

> db.restaurants.find({
  "borough" : "Manhattan",
  "grades.score" : { $lt : 10 }
},
  {"name" : 1, "_id" : 0, "grades.score" : 1}
)
< { grades: [ { score: 2 }, { score: 11 }, { score: 12 }, { score: 12 } ],
  name: 'Dj Reynolds Pub And Restaurant' }
{ grades: [ { score: 3 }, { score: 4 }, { score: 6 }, { score: 0 } ],
  name: '1 East 66Th Street Kitchen' }
{ grades:
  [ { score: 12 },
    { score: 16 },
    { score: 9 },
    { score: 13 },
    { score: 11 } ],
  name: 'Glorious Food' }
{ grades: [ { score: 12 }, { score: 11 }, { score: 6 }, { score: 8 } ],
  name: 'Bully\'S Deli' }
{ grades:
  [ { score: 10 },
```

10. Afficher les restaurants des Quartiers de NewYork de manière unique (Distinct). Les restaurants sont ceux de NewYork, il faut préciser l'attribut du Distinct (borough).

```
>_MONGOSH

> db.restaurants.distinct("borough")
< [
  'Bronx',
  'Brooklyn',
  'Manhattan',
  'Missing',
  'Queens',
  'Staten Island'
]
new_york> |
```

11. Donner la liste de valeurs comme des grades donnés par les inspecteurs en utilisant Distinct.

```
>_MONGOSH

> db.restaurants.distinct("grades.grade");
< [ 'A', 'B', 'C', 'Not Yet Graded', 'P', 'Z' ]
> db.restaurants.distinct("grades.score");
< [
  null, -1, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10,
  11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
  23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
  35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
  47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
  59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
  71, 73, 75, 76, 77, 78, 79, 80, 81, 82, 84, 86,
  89, 90, 92, 98, 131
]
new_york> |
```

12. Faites la même requête précédente en utilisant les agrégats (aggregate : match & project)

```
>_MONGOSH
> db.restaurants.aggregate( [
  { $match : { "grades.grade" : "C" } },
  { $project : { "name" : 1, "_id" : 0, "borough" : 1 } }
] )
< { borough: 'Brooklyn', name: 'May May Kitchen' }
{ borough: 'Queens', name: 'Terminal Cafe/Yankee Clipper' }
{ borough: 'Staten Island', name: 'B & M Hot Bagel & Grocery' }
{ borough: 'Manhattan', name: 'Texas Rotisserie' }
{ borough: 'Manhattan', name: 'Old Town Bar & Restaurant' }
{ borough: 'Brooklyn', name: 'Polish National Home' }
{ borough: 'Manhattan', name: 'Nyac Main Dining Room' }
{ borough: 'Manhattan', name: 'Marchis Restaurant' }
{ borough: 'Manhattan', name: 'Tout Va Bien' }
{ borough: 'Manhattan', name: 'Como Pizza' }
{ borough: 'Brooklyn', name: 'Shell Lanes' }
{ borough: 'Manhattan', name: 'Reynold\'S Bar' }
{ borough: 'Bronx', name: 'Yankee Tavern' }
{ borough: 'Queens', name: 'King Yum Restaurant' }
{ borough: 'Manhattan', name: 'The Princeton Club' }
{ borough: 'Manhattan', name: 'Nanni Restaurant' }
{ borough: 'Queens', name: 'Don Peppe' }
```

13. Ajouter un commentaire sur un restaurant (opération \$set) :

```
>_MONGOSH
> db.restaurants.update (
  {"_id" : ObjectId("64557377a2118f5952b5116c")},
  {$set : {"comment" : "My new comment"}}
);
< { acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
new_york>
```

```
< { _id: ObjectId("64557377a2118f5952b5116c"),
  address:
    { building: '1028',
      coord: { type: 'Point', coordinates: [ -73.966032, 40.762832 ] },
      street: '3 Avenue',
      zipcode: '10065' },
  borough: 'Manhattan',
  cuisine: 'Italian',
  grades:
    [ { date: 2014-09-16T00:00:00.000Z, grade: 'A', score: 13 },
      { date: 2014-02-24T00:00:00.000Z, grade: 'A', score: 10 },
      { date: 2013-05-03T00:00:00.000Z, grade: 'A', score: 10 },
      { date: 2012-08-20T00:00:00.000Z, grade: 'A', score: 7 },
      { date: 2012-02-13T00:00:00.000Z, grade: 'A', score: 9 } ],
  name: 'Isle Of Capri Restaurant',
  restaurant_id: '40364373',
  comment: 'My new comment' }
new_york>
```

14. Pour supprimer une clé, il suffit de remplacer par l'opérateur \$unset.

```
>_MONGOSH

> db.restaurants.update (
  {"_id" : ObjectId("64557377a2118f5952b5116c")},
  {$unset : {"comment" : 1}}
);
< { acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
new_york>
```

15. Attribuer un commentaire en fonction des grades obtenus. S'il n'a pas eu de note 'C', nous rajouterons le commentaire 'acceptable'.

UNE SEULE

```
>_MONGOSH

> db.restaurants.update (
  {"grades.grade" : {$not : {$eq : "C"}}},
  {$set : {"comment" : "acceptable"}}
);
< { acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
new_york>
```

16. Ajouter 3 points pour un rang A, 1 point pour B, et -1 pour C. Puis, montrer la meilleure note est obtenue

```
>_MONGOSH

> db.collection.aggregate([
  {
    $addFields: {
      total_points: {
        $sum: {
          $switch: {
            branches: [
              {
                case: { $eq: ["$grades.grade", "A"] },
                then: 3
              },
              {
                case: { $eq: ["$grades.grade", "B"] },
                then: 1
              },
              {
                case: { $eq: ["$grades.grade", "C"] },
                then: -1
              }
            ],
            default: 0
          }
        }
      }
    }
  }
])
```



```
    },
    {
      $project: {
        _id: 1,
        restaurant_id: 1,
        name: 1,
        borough: 1,
        cuisine: 1,
        address: 1,
        total_points: 1
      }
    },
    {
      $sort: {
        total_points: -1
      }
    },
    {
      $limit: 1
    }
  ]);
<
new_york>
```

17. Enlever les restaurants ayant la note 0.

```
> _MONGOSH
> db.restaurants.remove(
  {"note":0},
  {"multi" : true}
);
< { acknowledged: true, deletedCount: 1 }
new_york>
```

18. Exécuter le code suivant. Commenter.

```
mapFunction = function () {  
  emit(this.borough, 1);  
}  
reduceFunction = function (key, values) {  
  return Array.sum(values);  
}  
queryParam = {"query": {}, "out": {"inline": true}};  
db.restaurants.mapReduce(mapFunction, reduceFunction, queryParam);
```

```
> _MONGOSH  
  
> mapFunction = function () {  
  emit(this.borough, 1);  
  reduceFunction = function (key, values) {  
    return Array.sum(values);  
  };  
  queryParam = {"query": {}, "out": {"inline": true}};  
  db.restaurants.mapReduce(mapFunction, reduceFunction, queryParam);  
< {  
  results: [  
    { _id: 'Bronx', value: 2338 },  
    { _id: 'Brooklyn', value: 6085 },  
    { _id: 'Manhattan', value: 10257 },  
    { _id: 'Missing', value: 51 },  
    { _id: 'Queens', value: 5656 },  
    { _id: 'Staten Island', value: 969 }  
  ],  
  timeMillis: 699,  
  counts: { input: 25356, emit: 25356, reduce: 1286, output: 6 },  
  ok: 1  
}  
new_york>
```

Cette séquence de commandes calcule le nombre de documents dans la collection **restaurants** (et non pas paris) pour chaque valeur unique de l'attribut borough.

La sortie est une collection de paires clé-valeur, où la clé est la valeur de l'attribut borough et la valeur est le nombre de documents pour cette valeur de borough.