



INDEXATION DES DONNEES

RAPPORT EXPLICATIF DES SOLUTIONS CHOISIES



07 février 2019

Rédigé par Karim AMMOUR

École d'ingénieur ÉSTIAM Paris 20ème

Sommaire

INTRODUCTION	3
QUESTION 1	4
QUESTION 2	5
QUESTION 3	6
QUESTION 4	7
QUESTION 5	9
QUESTION 6	11
QUESTION 7	13
QUESTION 8	14
QUESTION 9	15

Introduction

Ce rapport a été rédigé dans le cadre du module d'indexation des données. Il constitue une aide pour la compréhension des différentes solutions que j'ai mis en œuvre pour répondre à un panel de questions.

Avant toutes choses, veuillez effectuer le mapping suivant afin que par la suite, vous puissiez exécuter l'ensemble des requêtes, sans générer d'erreur. (J'y reviendrai dès que nécessaire pour justifier ces choix.)

- 1) Mettre dans un fichier code.json le code suivant :

```
{
  "mappings" : {
    "_doc" : {
      "properties" : {
        "address.coord" : {
          "type" : "geo_point"
        },
        "borough" : { "type" : "keyword"},
        "cuisine" : { "type" : "keyword"}
      }
    }
  }
}
```

- 2) Ouvrir un terminal puis lancer la commande suivante :

curl -XPUT -H "Content-Type: application/json" 'localhost:9200/restaurants' --data-binary @code.json
Qui sert à créer l'indice restaurants et d'y joindre le mapping adéquat en fonction des questions préétablies.

- 3) Charger le fichier restaurants via la commande

curl -H "Content-Type: application/x-ndjson" -XPOST 'localhost:9200/restaurants/_doc/_bulk?pretty' -data-binary @restaurants.json'

Question n°1 :

Pour cette question il m'a fallu mettre dans le fichier de mapping le champ cuisine en keyword, mais alors pourquoi cela ? Bien lorsque par défaut le champ est en « text » j'ai remarqué que :

Sans le mot clef keyword (text par défaut) :

Type de cuisine	Devient	Term « japanese »
« Chinese/Japanese »	« chinese/japanese »	VRAI
« Japanese Chinese »	« japanese » et « chinese »	VRAI et FAUX
« Japanese4 »	« japanese4 »	VRAI
« Japanese »	« japanese »	VRAI

On remarque ici que les types "Chinese/Japanese" et "Japanese" sont tous les deux confondus et indissociables.

Avec le mot clef keyword :

Type de cuisine	Devient	Term « Japanese »
« Chinese/Japanese »	« Chinese/Japanese »	FAUX
« Japanese Chinese »	« Japanese Chinese »	FAUX
« Japanese4 »	« Japanese4 »	FAUX
« Japanese »	« Japanese »	VRAI

Avec le mot clef keyword le ou les types de cuisine proposés par un restaurant sont fixe et dissociable les uns des autres. (À partir du moment où ils ont au moins un caractère qui diffèrent).

Ainsi, dès lors que j'utilise un term : "Japanese" il me sera retourné uniquement les cuisines de type "Japanese" (cf figure de droite).

```
total" : 81,
max_score" : 4.8925786,
hits" : [
  {
    "_index" : "restaurants",
    "_type" : "_doc",
    "_id" : "40374415",
    "_score" : 4.8925786,
    "_source" : {
      "address" : {
        "building" : "113",
        "coord" : [
          -74.0019729,
          40.7260047
        ],
        "street" : "Thompson",
        "zipcode" : "10012"
      },
      "borough" : "Manhattan",
      "cuisine" : "Japanese",
      "grades" : [
        "borou" : "Brooklyn",
        "cuisine" : "Chinese/Japanese"
      ]
    }
  }
]
```

```
total" : 80,
max_score" : 4.26469,
hits" : [
  {
    "_index" : "restaurants",
    "_type" : "_doc",
    "_id" : "40374415",
    "_score" : 4.26469,
    "_source" : {
      "address" : {
        "building" : "113",
        "coord" : [
          -74.0019729,
          40.7260047
        ],
        "street" : "Thompson",
        "zipcode" : "10012"
      },
      "borough" : "Manhattan",
      "cuisine" : "Japanese",
      "grades" : [
        "borou" : "Brooklyn",
        "cuisine" : "Chinese/Japanese"
      ]
    }
  }
]
```

Question n°2 :

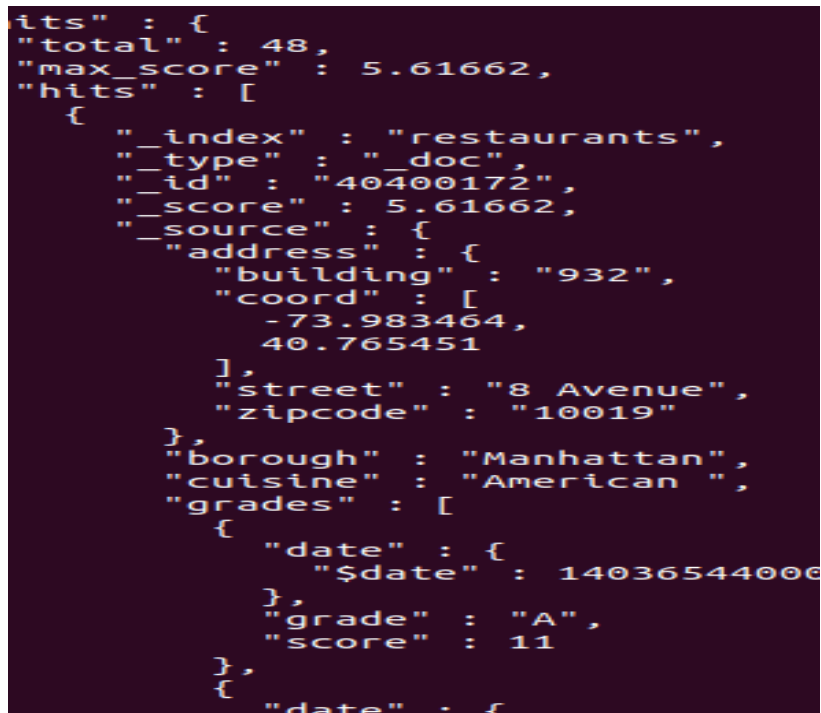
On prend ici l'ensemble des restaurants qui ont pour adresse "8 Avenue" le and est indispensable pour associer les mots "8" et "Avenue" par défaut c'est un or. Ce qui ne permet pas d'avoir la même sortie.

Code.json :

```
{
  "query": {
    "match": {
      "address.street": {
        "query": "8 avenue",
        "operator": "and"
      }
    }
  },
  "size" : 10000
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```



```
its" : {
  "total" : 48,
  "max_score" : 5.61662,
  "hits" : [
    {
      "_index" : "restaurants",
      "_type" : "_doc",
      "_id" : "40400172",
      "_score" : 5.61662,
      "_source" : {
        "address" : {
          "building" : "932",
          "coord" : [
            -73.983464,
            40.765451
          ],
          "street" : "8 Avenue",
          "zipcode" : "10019"
        },
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "grades" : [
          {
            "date" : {
              "$date" : 14036544000
            },
            "grade" : "A",
            "score" : 11
          },
          {
            "date" : {
```

Question n°3 :

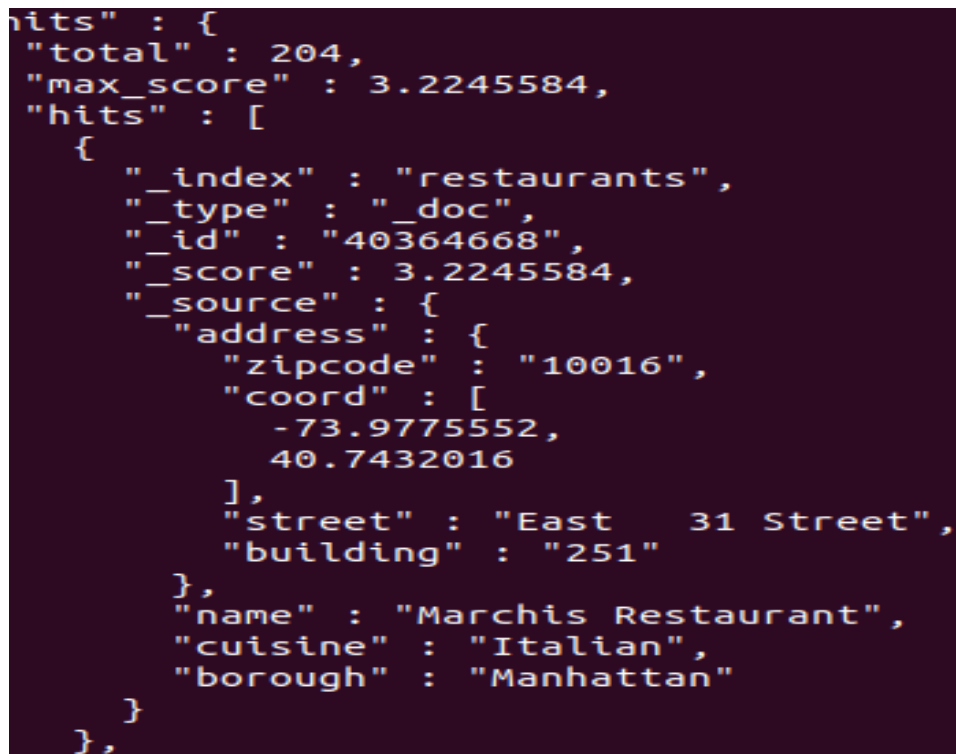
On prend ici l'ensemble des restaurants pour lesquels le type de cuisine doit être "Italian" et le type de quartier doit être "Manhattan", donc les 2 conditions à la fois. (Par le biais du bool, must, match) Pour plus d'ergonomie j'affiche seulement les champs les plus utiles.

Code.json :

```
{
  "query": {
    "bool": {
      "must": [
        { "match": { "cuisine": "Italian" } },
        { "match": { "borough": "Manhattan" } }
      ]
    }
  },
  "_source": ["cuisine", "borough", "address.*", "name"],
  "size" : 10000
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```



```
hits" : {
  "total" : 204,
  "max_score" : 3.2245584,
  "hits" : [
    {
      "_index" : "restaurants",
      "_type" : "_doc",
      "_id" : "40364668",
      "_score" : 3.2245584,
      "_source" : {
        "address" : {
          "zipcode" : "10016",
          "coord" : [
            -73.9775552,
            40.7432016
          ],
          "street" : "East 31 Street",
          "building" : "251"
        },
        "name" : "Marchis Restaurant",
        "cuisine" : "Italian",
        "borough" : "Manhattan"
      }
    },
    ...
  ]
}
```

Question n°4 :

On prend ici l'ensemble des restaurants qui ont pour type de cuisine "Hamburgers", et, parmi eux je regarde ceux qui sont dans un rayon de 2km, à partir des coordonnées GPS de l'Union Square qui d'après le fichier restaurant sont "lat" : 40.7348529, "lon" : -73.9897794 .

Pour être sûr à 100% j'ai pris le soin de regarder sur Google Maps si chacun des restaurants était bien dans la zone souhaitée (c'est bien le cas).

Code.json :

```
{
  "_source": ["cuisine", "borough", "address.*", "name"],
  "size" : 10000,
  "query": {
    "bool": {
      "must": [
        {"match" : { "cuisine": "Hamburgers" }}
      ],
      "filter": {
        "geo_distance" : {
          "distance" : "2km",
          "address.coord" : {
            "lat" : 40.7348529,
            "lon" : -73.9897794
          }
        }
      }
    }
  }
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```

Ci-après je montre le cas d'un exemple concret :

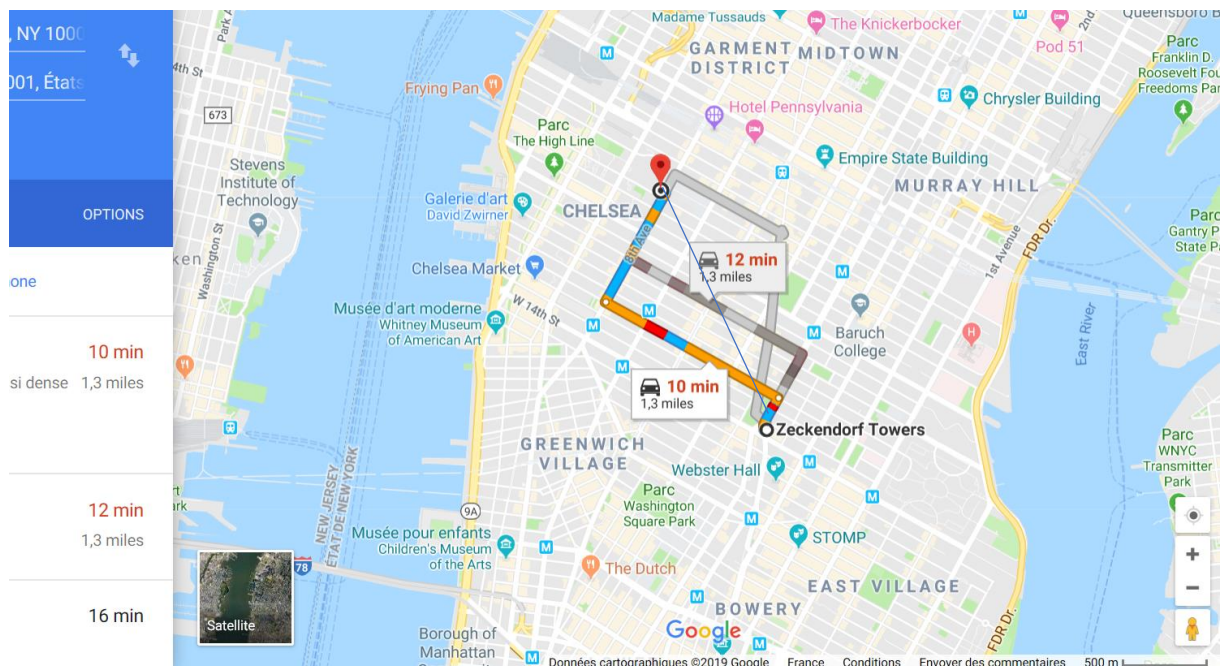
Il s'agit du restaurant Mcdonald'S à Manhattan dont les coordonnées GPS sont :

"lat" : -73.9972555

"lon" : 40.7476408

QUESTION N°4 : Où manger un hamburger proche de chez moi ? J'habite à côté de Union Square

```
"hits" : {  
  "total" : 8,  
  "max_score" : 3.50255,  
  "hits" : [  
    {  
      "_index" : "restaurants",  
      "_type" : "_doc",  
      "_id" : "40393474",  
      "_score" : 3.50255,  
      "_source" : {  
        "address" : {  
          "zipcode" : "10001",  
          "coord" : [  
            -73.9972555,  
            40.7476408  
          ],  
          "street" : "8 Avenue",  
          "building" : "335"  
        },  
        "name" : "McDonald's",  
        "cuisine" : "Hamburgers",  
        "borough" : "Manhattan"  
      }  
    },  
    ...  
  ]  
}
```



Comme l'illustre cette capture le chemin proposé par google montre une distance de 1.3 mile (environ deux km). Cela étant le segment (en bleu) qui relie les points de départ et d'arriver et inférieur 2km.

Question n°5 :

Celle-ci est très similaire à la précédente mise à part que l'on doit regarder non pas dans un rayon à partir d'un point, mais à l'intérieur d'un cercle délimité par un certain nombre de points.

On prend ici l'ensemble des restaurants qui ont pour type de cuisine "Pizza" et parmi eux je regarde ceux qui sont situés à l'intérieur du polygone issues des différents points de coordonnées GPS saisies.

J'ai également pris le soin de regarder quelques résultats pour voir s'ils se situaient bien dans le polygone en question (c'est bien le cas).

Code.json :

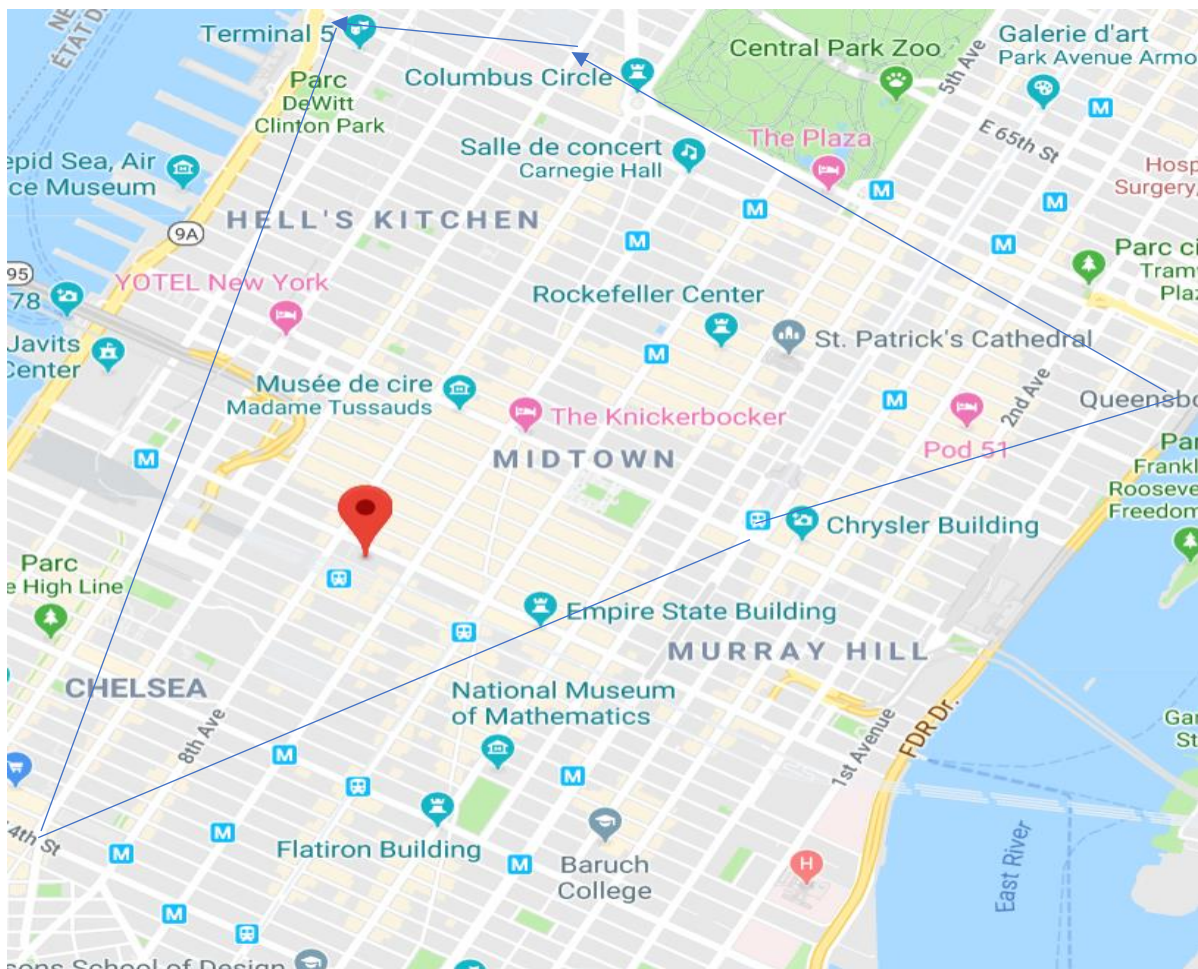
```
{
  "_source": ["cuisine", "borough", "address.*", "name"],
  "size" : 10000,
  "query": {
    "bool" : {
      "must" : {
        "match" : { "cuisine": "Pizza" }
      },
      "filter" : {
        "geo_polygon" : {
          "address.coord" : {
            "points" : [
              {"lat" : "40.769202", "lon" : "-73.984710"},
              {"lat" : "40.771418", "lon" : "-73.994003"},
              {"lat" : "40.741077", "lon" : "-74.005066"},
              {"lat" : "40.752430", "lon" : "-73.978397"},
              {"lat" : "40.757389", "lon" : "-73.960824"}
            ]
          }
        }
      }
    }
  }
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```

QUESTION N°5 : La liste des pizzerias situées entre le coin de la 9e ave et la 57e rue et la Gare Centrale

```
its" : {  
  "total" : 16,  
  "max_score" : 2.8848696,  
  "hits" : [  
    {  
      "_index" : "restaurants",  
      "_type" : "_doc",  
      "_id" : "40560383",  
      "_score" : 2.8848696,  
      "_source" : {  
        "address" : {  
          "zipcode" : "10119",  
          "coord" : [  
            -73.992471,  
            40.7511286  
          ],  
          "street" : "Penn Plaza",  
          "building" : "1"  
        },  
        "name" : "Rose'S Pizza",  
        "cuisine" : "Pizza",  
        "borough" : "Manhattan"  
      }  
    },  
    ]  
  }  
}
```



On peut bien voir que la pizzéria en question se situe à l'intérieur du polygone prédéfini (en bleu).

Question n°6 :

Pour cette question j'ai dû mettre en keyword le champ borough. Comme expliqué auparavant si je ne le fait pas il dissocie certains types de quartier par exemple borough : "Staten Island" est séparé en deux types de quartier distinct qui sont "Staten" et "Island". Avec le mot clé keyword je fixe le type du champ ainsi "Staten Island" reste inchangé.

Avec le "terms" je compte ici le nombre de restaurant par type de quartier

Code.json :

```
{
  "aggs": {
    "boroughs": {
      "terms": { "field": "borough", "size": 10000 }
    }
  },
  "size": 0
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```

```
"hits" : {
  "total" : 3772,
  "max_score" : 0.0,
  "hits" : [ ]
},
"aggregations" : {
  "boroughs" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "Manhattan",
        "doc_count" : 1883
      },
      {
        "key" : "Queens",
        "doc_count" : 738
      },
      {
        "key" : "Brooklyn",
        "doc_count" : 684
      },
      {
        "key" : "Bronx",
        "doc_count" : 309
      },
      {
        "key" : "Staten Island",
        "doc_count" : 158
      }
    ]
  }
}
```

Maintenant voici la preuve qu'avec le type text par défaut ça ne fonctionne pas correctement.

- 1) Effectuer le mapping sans le mot clé keyword pour le champ borough
- 2) Changer le fielddata du champ borough en true afin de pouvoir faire une agrégation sur un champ de type text (par défaut)

Code.json :

```
{
  "properties" : {
    "borough" : {
      "type" : "text",
      "fielddata" : true
    }
  }
}
```

- 3) Entrer la commande suivante pour que le changement soit effective :

```
curl -XPUT -H "Content-Type: application/json" 'localhost:9200/restaurants/_mapping/_doc' --data-binary '@code.json'
```

- 4) Recharger le fichier code.json

Code.json :

```
{
  "aggs" : {
    "boroughs" : {
      "terms" : { "field" : "borough" , "size" : 10000}
    }
  },
  "size": 0
}
```

On peut voir la dissociation de "Staten Island" en "Staten" et "Island"

```
{
  "borough" : {
    "type" : "text",
    "fields" : {
      "keyword" : {
        "type" : "keyword",
        "ignore_above" : 256
      }
    }
  },
  "fielddata" : true
},
```

```
{
  "key" : "brooklyn",
  "doc_count" : 684
},
{
  "key" : "bronx",
  "doc_count" : 309
},
{
  "key" : "island",
  "doc_count" : 158
},
{
  "key" : "staten",
  "doc_count" : 158
}
]
```

Question n°7 :

Encore une fois dans le mapping pour le champ cuisine le type keyword et nécessaire pour les mêmes raisons

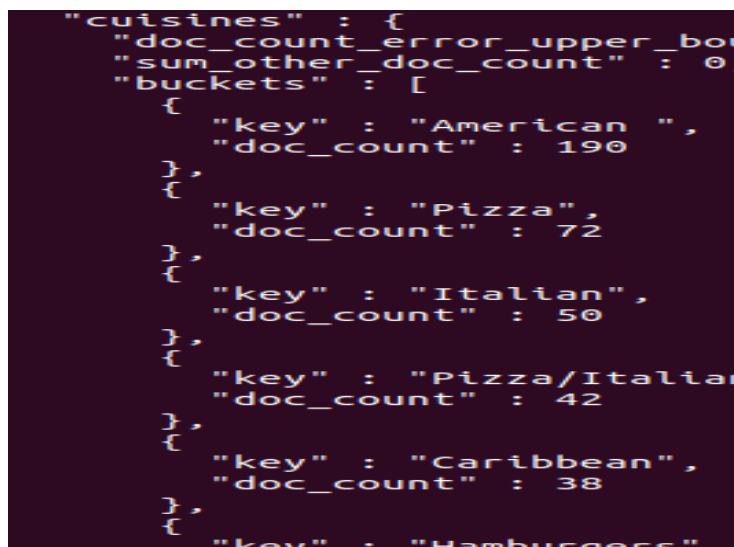
On prend ici l'ensemble des restaurants situés à Brooklyn dans lesquels je compte grâce au terms le nombre de restaurants par type de cuisine (à Brooklyn encore une fois)

Code.json :

```
{
  "size" : 1,
  "query": {
    "bool": {
      "must": [
        {"match" : { "borough": "Brooklyn" }}
      ]
    }
  },
  "aggs" : {
    "cuisines" : {
      "terms" : { "field" : "cuisine", "size" : 10000}
    }
  }
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```



```
"cuisines" : {
  "doc_count_error_upper_bound" : 0,
  "sum_other_doc_count" : 0,
  "buckets" : [
    {
      "key" : "American ",
      "doc_count" : 190
    },
    {
      "key" : "Pizza",
      "doc_count" : 72
    },
    {
      "key" : "Italian",
      "doc_count" : 50
    },
    {
      "key" : "Pizza/Italian",
      "doc_count" : 42
    },
    {
      "key" : "Caribbean",
      "doc_count" : 38
    },
    {
      "key" : "Hamburgers"
```

Question n°8 :

Encore une fois dans le mapping pour le champ « name » le type keyword et nécessaire pour les mêmes raisons. Ici je prends l'ensemble des restaurants Italiens de Manhattan pour lesquels j'effectue un ensemble d'agrégation.

Tout d'abord, j'effectue une dissociation de chaque restaurant de Manhattan qui ont pour type de cuisine Italian (restaurantMI). Ensuite, j'ordonne les résultats qui sont calculés dans l'agrégation « avg_score » par ordre décroissant, ainsi la size = 1 n'affichera que la plus grande value.

Avg_score calcule une moyenne des scores (A, B, C) pour chacun des restaurants (ici Italien de Manhattan).

Code.json :

```
{
  "size" : 0,
  "query": {
    "bool": {
      "must": [
        { "match": { "cuisine": "Italian" }},
        { "match": { "borough": "Manhattan" }}
      ]
    }
  },
  "aggs" : {
    "restaurantsMI" : {
      "terms" : {
        "field" : "name",
        "order": {
          "avg_score" : "desc"
        },
        "size": 1
      },
      "aggs": {
        "avg_score": {
          "avg": { "field": "grades.score" }
        }
      }
    }
  }
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```

```
{
  "took" : 95,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 204,
    "max_score" : 0.0,
    "hits" : [ ]
  },
  "aggregations" : {
    "restaurantsMI" : {
      "doc_count_error_upper_bound" : -1,
      "sum_other_doc_count" : 203,
      "buckets" : [
        {
          "key" : "Nanni Restaurant",
          "doc_count" : 1,
          "avg_score" : {
            "value" : 32.142857142857146
          }
        }
      ]
    }
  }
}
```

Question n°9 :

On prend l'ensemble des restaurants de Manhattan pour lesquels j'effectue un ensemble d'agrégation.

Tout d'abord, j'effectue une dissociation de chaque type de cuisine de Manhattan. Et pour chacun des types de cuisine qui existe à Manhattan j'effectue la même chose que la question précédente.

Code.json :

```
{
  "size" : 0,
  "query": {
    "bool": {
      "must": [
        { "match": { "borough": "Manhattan" } }
      ]
    }
  },
  "aggs" : {
    "cuisines" : {
      "terms" : { "field" : "cuisine", "size" : 10000}
    },
    "aggs" : {
      "restaurantsMI" : {
        "terms" : {
          "field" : "name",
          "order": {
            "avg_score" : "desc"
          },
          "size": 1
        },
        "aggs": {
          "avg_score": {
            "avg": { "field": "grades.score" }
          }
        }
      }
    }
  }
}
```

Entrer la commande :

```
curl -XPOST -H "Content-Type: application/json" 'localhost:9200/restaurants/_search?pretty' --data-binary '@code.json'
```



```
"doc_count_error_upper_bound" : -1,
"sum_other_doc_count" : 733,
"buckets" : [
  {
    "key" : "West 79Th Street Boat Basin Cafe",
    "doc_count" : 1,
    "avg_score" : {
      "value" : 36.0
    }
  }
]
},
{
  "key" : "Italian",
  "doc_count" : 204,
  "restaurantsMI" : {
    "doc_count_error_upper_bound" : -1,
    "sum_other_doc_count" : 203,
    "buckets" : [
      {
        "key" : "Nanni Restaurant",
        "doc_count" : 1,
        "avg_score" : {
          "value" : 32.142857142857146
        }
      }
    ]
  }
},
{
  "key" : "Café/Coffee/Tea",
  "doc_count" : 133,
  "restaurantsMI" : {
    "doc_count_error_upper_bound" : -1,
    "sum_other_doc_count" : 132,
    "buckets" : [
      {
        "key" : "World Cup Cafe",
        "doc_count" : 1,
        "avg_score" : {
          "value" : 17.375
        }
      }
    ]
  }
},
{
```