BUT 3 Science des données
Compétence 1 : Traiter des données à des fins décisionnelles

Semestre 5 – Bases de données NoSQL (30h)

Typologie NoSQL (13,5h) – S.Abboud

Python MongoDB (16,5h) – F.Garnier

BUT 3 Science des données
Compétence 1 : Traiter des données à des fins décisionnelles

Progression proposée (F.Garnier)

Chapitre 1 : **Rappel langage Python**

Chapitre 2 : **Cartographie**

Chapitre 3 : **Python et MongoDB**

Chapitre 4 : **Programmation objet**

- 1,5 h CM
- 15 h TD

BUT Science des données Niort

Chap. 3 – Python et MongoDB

Objectif : Récupération de données dénormalisées dans une base NoSQL : `MongoDB`, SGBD orienté "Documents"

Packages utilisés: `PyMongo`

https://api.mongodb.com/python/current/
https://www.mongodb.com

Installation de MongoDB :

https://www.mongodb.com/try/download/community
https://docs.mongodb.com/manual/installation/

https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4474601-decouvrez-le-fonctionnement-de-mongodb

Chap. 3 – Python et MongoDB

Présentation

`MongoDB` est un SGBD NoSQL de type *Document Store* (orienté document)
www.mongodb.com

Objectifs :

- Gérer de gros volumes
- Facilité de déploiement et d'utilisation
- Possibilité de faire des traitements complexes

Modèle de données

Principe de base : Les données sont des documents.

Pas de schéma des documents définis en amont *(pas de schéma de données contrairement à une BD relationnelle ou NoSQL de type Column Store)*

Les documents :

- sont stockés en *Binary JSON* (BSON)
- sont similaires et rassemblés dans des collections
- peuvent n'avoir aucun point commun entre eux
- contiennent (généralement) l'ensemble des informations (*pas ou très peu de jointures à faire*)
- BD respectant **CP** (dans le théorème *CAP*) – propriétés ACID au niveau d'un document

Exemple Passage MCD -> JSON

Format JSON

- JavaScript Object Notation, créé en 2005
- Format **léger** d'échange de données structurées (littéral)
- Schéma des données non connu (contenu dans les données)
- Basé sur une collection de couples clé/valeur ou une liste de valeurs ordonnées
- Structures possibles :
  - objet (couples clé/valeur) : {}
    Exemple : { "nom": "garnier", "prenom": "françois" }
  - tableau (liste de valeurs) : []
    Exemple : [1, 5, 10]
  - une valeur dans un objet ou dans un tableau peut être elle-même un littéral
- Deux types atomiques (`string` et `number`) et trois constantes (`true`, `false`, `null`)
- Validation possible du JSON sur jsonlint.com/

Exemple JSON

formations.json

```
{
"cnam": {
        "formation": "Ingénieur.e Big Data & IA",
        "responsable": { "nom": "Bartholme", "prenom": "Paul" },
        "etudiants" : [
                        { "id": 1, "nom": "Caisson", "prenom": "Thibaud" },
                        { "id": 2, "nom": "Madier", "details": "délégué" },
                        { "id": 5, "nom": "Leymarie" }
                        ],
        "ouverte": true
},
"sd": {
        "formation": "BUT Science des données",
        "ouverte": false,
        "todo": ["Creation de la maquette","Validation par le conseil"],
        "responsable": { "nom": "Garnier" }
        }
}
```

Format BSON

BSON : extension de JSON, format binaire

- Quelques types supplémentaires (identifiant spécifique, binaire, date, ...)
- Distinction entier et réel
- **Schéma dynamique**
  - Documents variants très fortement entre eux, même dans une même collection
  - On parle de **self-describing documents**
  - Ajout très facile d'un nouvel élément pour un document, même si cet élément est inexistant pour les autres
  - Pas de ALTER TABLE ou de redesign de la base
- **Pas de jointures entre les collections**
- **Langage d'interrogation (JSON, BSON) :**
  - Pas de SQL (bien évidemment), ni de langage proche
  - Définition d'un langage propre basé sur javascript

Rappel Théorème CAP

Proposé par Brewer (2000), puis amélioré par la suite par Gilbert et Lynch.

Il existe trois propriétés essentielles d'un système :

- **Consistency** (cohérence) : les données sont cohérentes entre tous les noeuds
- **Availability** (disponibilité) : les données sont disponibles à n'importe quel moment
- **Partition Tolerance** (resistance au partitionnement) : le système continue de fonctionner même si un des noeuds est inopérant.

Problème : Aucun système distribué ne peut respecter ces trois propriétés.
On a donc le choix entre :
C + A : un problème sur un des nœuds fait stopper le système (les SGBDR classiques sont plutôt dans cette catégorie)
C + P : les données ne sont pas forcément disponibles au moment de la requête
A + P : les données renvoyées ne sont pas toujours cohérentes

Rappel Théorème CAP (suite)

Conséquence du passage à l'échelle
Le passage à l'échelle implique (presque obligatoirement) le *partitionnement* des données
Il faut donc faire le choix entre *cohérence* et *disponibilité*
Dans pratiquement tous les systèmes, la disponibilité est préférée, et donc la cohérence stricte est abandonnée (d'où le non-respect de *ACID*)

Heureusement, une réponse existe - **BASE** :

- **Basically Available** : il y aura une réponse à toute requête, même si c'est du genre *failure* ou *inconsistent data*
- **Soft State** : le système n'est pas consistent à tout instant
- **Eventually consistent** : le système deviendra finalement consistent, lorsqu'il ne recevra plus d'entrées

Tous les systèmes actuels des géants du web sont dans cette configuration **BASE**

Chap. 3 – Python et MongoDB

Installation de MongoDB

- Lancer une connexion au serveur mongo via l'interface graphique "mongodb compass" (à installer si pas présent)
- Créer une database **resto** puis une collection **restaurants**
- En mode cmd, importer le fichier bson (pas possible dans compass uniquement json ou csv) : `mongorestore –d resto –c restaurants d:\mongodbdata\4-restaurants.bson`
- Vérifier sous compass que les documents ont été importés.

Si la connexion au server mongo sur le port 27017 est impossible c'est sans doute que l'espace de stockage des datas n'existe pas (par défaut `c:\data\mongo…`)

- Créer donc un espace de stockage des datas, par exemple `d:\mongodbdata`
- En mode cmd : `mongod –dbpath d:\mongodbdata`
- **Laisser mode cmd ouvert**

Pour que l'ensemble des commandes mongo soient reconnues, pensez à créer une nouvelle variable dans le path des variables systèmes de votre machine, contenant le répertoire `bin` de mongodb.

Chap. 3 – Python et MongoDB

Premiers scripts python

```
import pymongo
connex = pymongo.MongoClient("mongodb://127.0.0.1:27017/")
db = connex.resto

# paramètre {} obligatoire
print(db.restaurants.count_documents({}))

# A privilégier en en cas de multiples serveurs et de données massives
print(db.restaurants.estimated_document_count())
```

25359

```
# données Json = dictionnaire python
# Récupération du 1er document (un curseur ici)
print(db.restaurants.find(limit = 1))

# Pour visualiser le résultat : transtypage en list, DataFrame, …
premier = db.restaurants.find(limit = 1)
print(list(premier))
```

<pymongo.cursor.Cursor object at 0x000002BDFE13ECC8>

[{'_id': ObjectId('58ac16d1a251358ee4ee87dd'), 'address': {'building': '1007', 'coord': [-73.856077, 40.848447], 'street': 'Morris Park Ave', 'zipcode': '10462'}, 'borough': 'Bronx', 'cuisine': 'Bakery', 'grades': [{'date': datetime.datetime(2014, 3, 3, 0, 0), 'grade': 'A', 'score': 2}, {'date': datetime.datetime(2013, 9, 11, 0, 0), 'grade': 'A', 'score': 6}, {'date': datetime.datetime(2013, 1, 24, 0, 0), 'grade': 'A', 'score': 10}, {'date': datetime.datetime(2011, 11, 23, 0, 0), 'grade': 'A', 'score': 9}, {'date': datetime.datetime(2011, 3, 10, 0, 0), 'grade': 'B', 'score': 14}], 'name': 'Morris Park Bake Shop', 'restaurant_id': '30075445'}]

Comptage avec sélection de documents *(idem dans les fonctions **distinct** et **find** étudiées plus tard)*

Les paramètres suivants seront possibles pour la fonction **`count_documents()`**

- `{}` `# tous les documents`
- `{"champ":valeur}`
- `{condition1,condition2}` `# ET logique`
- `champ.sousChamp`
- `{"champ":{"$operateur":expression}}`
  - **`$operateur`** *peut être parmi :*
    - `$in`
    - `$gt, $gte, $lt, $lte, $ne`

  `# comparaison : greather than, greather than or equal, less than, …, not equal`

Chap. 3 – Python et MongoDB

Comptage avec sélection (exemples)

```
# Nombre de restaurants du quartier de Brooklyn
print(db.restaurants.count_documents({ "borough": "Brooklyn" }))

# Nombre de restaurants du quartier de Brooklyn, cuisine Française
print(db.restaurants.count_documents({ "borough": "Brooklyn", "cuisine": "French" }))

# Restaurants de Brooklyn proposant de la cuisine française ou italienne
print(db.restaurants.count_documents({
      "borough": "Brooklyn", "cuisine":{ "$in": ["French", "Italian"]} }))

# Restaurants situés sur Franklin Street
print(db.restaurants.count_documents({"address.street": "Franklin Street"}))

# Restaurants ayant eu un score de 0
print(db.restaurants.count_documents({"grades.score": 0}))
# Restaurants ayant eu un score <= 5
print(db.restaurants.count_documents({"grades.score":{"$lte": 5}}))
```

6086

54

246

25

1246

10650

Chap. 3 – Python et MongoDB

Doublons

```
print(db.restaurants.distinct("borough"))   # ou distinct(key="borough")
```

['Bronx', 'Brooklyn', 'Manhattan', 'Missing', 'Queens', 'Staten Island']

```
# Type de cuisine pour les restaurants de Brooklyn
print(db.restaurants.distinct(key="cuisine",query={"borough":"Brooklyn"}))
```

['Afghan', 'African', 'American', 'Armenian', 'Asian', 'Australian', 'Bagels/Pretzels', 'Bakery', 'Bangladeshi', 'Barbecue', 'Bottled beverages, including water, sodas, juices, etc.', 'Brazilian', 'Café/Coffee/Tea', 'Cajun', 'Caribbean', 'Chicken', 'Chilean', 'Chinese', 'Chinese/Cuban', 'Chinese/Japanese', 'Continental', 'Creole', 'Creole/Cajun', 'Czech', 'Delicatessen', 'Donuts', 'Eastern European', 'Egyptian', 'English', 'Ethiopian', 'Filipino', 'French', 'Fruits/Vegetables', 'German', 'Greek', 'Hamburgers', 'Hawaiian', 'Hotdogs', 'Hotdogs/Pretzels', 'Ice Cream, Gelato, Yogurt, Ices', 'Indian', 'Indonesian', 'Irish', 'Italian', 'Japanese', 'Jewish/Kosher', 'Juice, Smoothies, Fruit Salads', 'Korean', 'Latin (Cuban, Dominican, Puerto Rican, South & Central American)', 'Mediterranean', 'Mexican', 'Middle Eastern', 'Moroccan', 'Not Listed/Not Applicable', 'Nuts/Confectionary', 'Other', 'Pakistani', 'Pancakes/Waffles', 'Peruvian', 'Pizza', 'Pizza/Italian', 'Polish', 'Portuguese', 'Russian', 'Salads', 'Sandwiches', 'Sandwiches/Salads/Mixed Buffet', 'Scandinavian', 'Seafood', 'Soul Food', 'Soups & Sandwiches', 'Southwestern', 'Spanish', 'Steak', 'Tapas', 'Tex-Mex', 'Thai', 'Turkish', 'Vegetarian', 'Vietnamese/Cambodian/Malaysia']]

```
# Grades des restaurants de Brooklyn
print(db.restaurants.distinct(key="grades.grade",query={"borough":"Brooklyn"}))
```

['A', 'B', 'C', 'Not Yet Graded', 'P', 'Z']

Projection de champs

La fonction `find` réalise des sélections (idem comptage) mais également des projections (affichage). Les principaux paramètres de la fonction sont les suivants :

- Sélection des documents
- **Projection des champs (champs à afficher en résultat de la requête)**
- `limit` pour n'avoir que les `n` premiers documents
- `sort` pour trier les documents en sortie
- `skip` ne considère pas les `n` premiers documents

Rappel : cette fonction renvoie un curseur à gérer pour l'affichage (**transtypage en list, DataFrame, …**)

Syntaxe pour les champs à afficher

Le 2ème paramètre de la fonction `find` va donc permettre de choisir les champs à afficher (projection) avec les critères suivants :

- Sans précision, l'identifiant est toujours affiché `(_id)`
- `"champ":1` => champ affiché
- `"champ":0` => champ non affiché

Remarque : ne pas mentionner les champs à 0 sauf pour l'identifiant interne à Mongo `(_id)`, ou alors ne mentionner que des champs à 0 ou ne mentionner que des champs à 1 !

Exemple : `{"_id":0,"champ":1,…}`

Chap. 3 – Python et MongoDB

Tri

Le paramètre `sort` de la fonction `find` prend **1 liste (ou tuple) composée** de un ou plusieurs tuples indiquant les critères de tri :

- `("champ":1)` => tri croissant
- `("champ":-1)` => tri décroissant

Remarque : plusieurs critères de tri peuvent être combinés

Chap. 3 – Python et MongoDB

Limit

```
print(pandas.DataFrame(db.restaurants.find(limit = 5)))
```

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	[{'date': 2014-03-03 00:00:00, 'grade': 'A', '...	Morris Park Bake Shop	30075445
1	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	[{'date': 2014-12-30 00:00:00, 'grade': 'A', '...	Wendy'S	30112340
2	60006d6aa7aafd5a6d45ca9c	{'building': '351', 'coord': [-73.985135599999...	Manhattan	Irish	[{'date': 2014-09-06 00:00:00, 'grade': 'A', '...	Dj Reynolds Pub And Restaurant	30191841
3	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018
4	60006d6aa7aafd5a6d45ca9e	{'building': '97-22', 'coord': [-73.8601152, 4...	Queens	Jewish/Kosher	[{'date': 2014-11-24 00:00:00, 'grade': 'Z', '...	Tov Kosher Kitchen	40356068

Exemples de recherche

```
print(pandas.DataFrame(db.restaurants.find({ "name": "Shake Shack" },
                                           { "address.street": 1, "borough": 1 })))
```

	_id	address	borough
0	60006d6ca7aafd5a6d45ec73	{'street': 'Columbus Avenue'}	Manhattan
1	60006d6ca7aafd5a6d45f66d	{'street': 'West 44 Street'}	Manhattan
2	60006d6ca7aafd5a6d45f66e	{'street': 'East 86 Street'}	Manhattan
3	60006d6ca7aafd5a6d45fe35	{'street': 'North End Avenue'}	Manhattan
4	60006d6ca7aafd5a6d45ff7f	{'street': 'Fulton Street'}	Brooklyn
5	60006d6da7aafd5a6d461437	{'street': 'Jfk International Airport'}	Queens
6	60006d6da7aafd5a6d4618f1	{'street': 'Grand Central Terminal'}	Manhattan
7	60006d6da7aafd5a6d46197c	{'street': 'Jfk International Airport'}	Queens
8	60006d6ea7aafd5a6d462178	{'street': 'Old Fulton Street'}	Brooklyn
9	60006d6ea7aafd5a6d46217a	{'street': 'Flatbush Avenue'}	Brooklyn
10	60006d6ea7aafd5a6d462955	{'street': '3Rd Ave'}	Manhattan

```
# Affichage sans l'identifiant interne
{ "_id": 0, "address.street": 1,
            "borough": 1 }
```

Chap. 3 – Python et MongoDB

Exemples de recherche

```
# 5 premiers restaurants du quartier Queens, avec une note A et un score supérieur à 50
# On souhaite le nom et la rue du restaurant.
print(pandas.DataFrame(db.restaurants.find(
                {"borough": "Queens", "grades.score": { "$gte":  50}},
                {"_id": 0, "name": 1, "grades.score": 1, "address.street": 1},
                limit = 5
                )))
```

	address	grades	name
0	{'street': 'Horace Harding Boulevard'}	[{'score': 12}, {'score': 4}, {'score': 11}, {...	Richer'S Bakery
1	{'street': 'Bell Boulevard'}	[{'score': 52}, {'score': 12}, {'score': 22}, ...	Tequilla Sunrise
2	{'street': 'Rockaway Beach Boulevard'}	[{'score': 10}, {'score': 2}, {'score': 10}, {...	Rockaway Beach Inn
3	{'street': 'Broadway'}	[{'score': 13}, {'score': 13}, {'score': 13}, ...	Alfonso'S Bar
4	{'street': 'Woodhaven Boulevard'}	[{'score': 2}, {'score': 64}, {'score': 9}, {'...	Pio Pio

Chap. 3 – Python et MongoDB

Exemples de recherche

```
# Restaurants Shake Shack dans différents quartiers (Queens et Brooklyn)
print(pandas.DataFrame(db.restaurants.find(
    {"name": "Shake Shack", "borough": {"$in": ["Queens", "Brooklyn"]}},
    {"_id": 0, "address.street": 1, "borough": 1}
      )))
```

	address	borough
0	{'street': 'Fulton Street'}	Brooklyn
1	{'street': 'Jfk International Airport'}	Queens
2	{'street': 'Jfk International Airport'}	Queens
3	{'street': 'Old Fulton Street'}	Brooklyn
4	{'street': 'Flatbush Avenue'}	Brooklyn

Exemples de recherche

```
# Restaurants du Queens ayant une note supérieure à 50, mais trié par ordre décroissant
# de noms de rue, et ordre croissant de noms de restaurants
print(pandas.DataFrame(db.restaurants.find(
    {"borough": "Queens", "grades.score": { "$gt":  50}},
    {"_id": 0, "name": 1, "address.street": 1},
    sort = (("address.street", -1), ("name", 1))
)))
```

	address	name
0	{'street': 'Woodward Avenue'}	Sabores Restaurant & Bar
1	{'street': 'Woodside Avenue'}	Salza Pizza
2	{'street': 'Woodside Avenue'}	Spicy Shallot
3	{'street': 'Woodhaven Boulevard'}	Fresh To You
4	{'street': 'Woodhaven Boulevard'}	Pio Pio
...	...	...
73	{'street': '30 Avenue'}	Queens Comfort Restaurant
74	{'street': '20 Avenue'}	Cafeteria (Usps Bldng)
75	{'street': '153 Avenue'}	Tuscany Deli
76	{'street': '131 Street'}	Spa Castle/Juice Farm
77	{'street': '102 Street'}	Tacos Mexico

Chap. 3 – Python et MongoDB

Cette fonctionnalité va permettre de prendre en paramètre un **pipeline** : tableau composé d'une suite d'opérations :

Fonction	Opération
`$limit`	restriction à un petit nombre de documents (très utiles pour tester son calcul)
`$sort`	tri sur les documents
`$match`	restriction sur les documents à utiliser
`$unwind`	séparation d'un document en plusieurs sur la base d'un tableau
`$addFields`	ajout d'un champs dans les documents
`$project`	redéfinition des documents
`$group`	regroupements et calculs d'aggégrats
`$sortByCount`	regroupement, calcul de dénombrement et tri déccroissant en une opération
`$lookup`	jointure avec une autre collection
...	

Chap. 3 – Python et MongoDB

Les opérations se font dans l'ordre d'écriture et le même opérateur peut donc apparaître plusieurs fois :

- `$limit` : un entier
- `$sort` : identique à celle du paramètre `sort` de la fonction `find()`
- `$match` : identique à celle du paramètre `query` des autres fonctions
- `$unwind` : nom du tableau servant de base pour le découpage (précédé d'un `$`)
    - un document avec un tableau à *n* éléments deviendra *n* documents avec chacun un des éléments du tableau en lieu et place de celui-ci
- `$sortByCount` : nom du champs sur lequel on veut le dénombrement et le tri décroissant selon le résultat

Chap. 3 – Python et MongoDB

Exemples

```
print(pandas.DataFrame(db.restaurants.aggregate(
    [{"$limit": 10}]
)))
```

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	[{'date': 2014-03-03 00:00:00, 'grade': 'A', '...	Morris Park Bake Shop	30075445
1	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	[{'date': 2014-12-30 00:00:00, 'grade': 'A', '...	Wendy'S	30112340
2	60006d6aa7aafd5a6d45ca9c	{'building': '351', 'coord': [-73.985135599999...	Manhattan	Irish	[{'date': 2014-09-06 00:00:00, 'grade': 'A', '...	Dj Reynolds Pub And Restaurant	30191841
3	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018
4	60006d6aa7aafd5a6d45ca9e	{'building': '97-22', 'coord': [-73.8601152, 4...	Queens	Jewish/Kosher	[{'date': 2014-11-24 00:00:00, 'grade': 'Z', '...	Tov Kosher Kitchen	40356068
5	60006d6aa7aafd5a6d45ca9f	{'building': '8825', 'coord':	Queens	American	[{'date': 2014-11-15	Brunos On The	40356151

Chap. 3 – Python et MongoDB

Exemples

```
print(pandas.DataFrame(db.restaurants.aggregate(
    [   { "$limit": 10 },
        { "$sort": { "name": 1 }}
    ]
)))
```

_id	address	borough	cuisine	grades	name	restaurant_id
60006d6aa7aafd5a6d45ca9f	{'building': '8825', 'coord': [-73.8803827, 40...	Queens	American	[{'date': 2014-11-15 00:00:00, 'grade': 'Z', '...	Brunos On The Boulevard	40356151
60006d6aa7aafd5a6d45ca9c	{'building': '351', 'coord': [-73.985135599999...	Manhattan	Irish	[{'date': 2014-09-06 00:00:00, 'grade': 'A', '...	Dj Reynolds Pub And Restaurant	30191841
60006d6aa7aafd5a6d45caa0	{'building': '2206', 'coord': [-74.1377286, 40...	Staten Island	Jewish/Kosher	[{'date': 2014-10-06 00:00:00, 'grade': 'A', '...	Kosher Island	40356442
60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	[{'date': 2014-03-03 00:00:00, 'grade': 'A', '...	Morris Park Bake Shop	30075445
60006d6aa7aafd5a6d45caa2	{'building': '6409', 'coord': [-74.00528899999...	Brooklyn	American	[{'date': 2014-07-18 00:00:00, 'grade': 'A', '...	Regina Caterers	40356649
60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord':	Brooklyn	American	[{'date': 2014-06-10	Riviera Caterer	40356018

Chap. 3 – Python et MongoDB

Exemples

```
# Restaurants par ordre alphabétique (il n'y en a que 5 dans les 10 !)
print(pandas.DataFrame(db.restaurants.aggregate(
    [    { "$limit": 10 },
         { "$sort": { "name": 1 }},
         { "$match": { "borough": "Brooklyn" }}
    ] )))
```

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45caa2	{'building': '6409', 'coord': [-74.00528899999...	Brooklyn	American	[{'date': 2014-07-18 00:00:00, 'grade': 'A', '...	Regina Caterers	40356649
1	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018
2	60006d6aa7aafd5a6d45caa3	{'building': '1839', 'coord': [-73.9482609, 40...	Brooklyn	Ice Cream, Gelato, Yogurt, Ices	[{'date': 2014-07-14 00:00:00, 'grade': 'A', '...	Taste The Tropics Ice Cream	40356731
3	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	[{'date': 2014-12-30 00:00:00, 'grade': 'A', '...	Wendy'S	30112340
4	60006d6aa7aafd5a6d45caa1	{'building': '7114', 'coord': [-73.9068506, 40...	Brooklyn	Delicatessen	[{'date': 2014-05-29 00:00:00, 'grade': 'A', '...	Wilken'S Fine Food	40356483

Chap. 3 – Python et MongoDB

Exemples

```
# Restaurants par ordre alphabétique (sélection avant la limite donc 10 résultats !)
print(pandas.DataFrame(db.restaurants.aggregate(
    [    { "$match": { "borough": "Brooklyn" }},
         { "$limit": 10 },
         { "$sort": { "name": 1 }}
    ])))
```

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45caa5	{'building': '7715', 'coord': [-73.9973325, 40...	Brooklyn	American	[{'date': 2014-04-16 00:00:00, 'grade': 'A', '...	C & C Catering Service	40357437
1	60006d6aa7aafd5a6d45caa9	{'building': '203', 'coord': [-73.978220400000...	Brooklyn	Ice Cream, Gelato, Yogurt, Ices	[{'date': 2014-02-10 00:00:00, 'grade': 'A', '...	Carvel Ice Cream	40360076
2	60006d6aa7aafd5a6d45caa6	{'building': '1269', 'coord': [-73.871194, 40....	Brooklyn	Chinese	[{'date': 2014-09-16 00:00:00, 'grade': 'B', '...	May May Kitchen	40358429
3	60006d6aa7aafd5a6d45caab	{'building': '6909', 'coord': [-74.0259567, 40...	Brooklyn	Delicatessen	[{'date': 2014-08-21 00:00:00, 'grade': 'A', '...	Nordic Delicacies	40361390
4	60006d6aa7aafd5a6d45caa2	{'building': '6409', 'coord': [-74.00528899999...	Brooklyn	American	[{'date': 2014-07-18 00:00:00, 'grade': 'A', '...	Regina Caterers	40356649
5	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018
6	60006d6aa7aafd5a6d45caa8	{'building': '705', 'coord': [	Brooklyn	Jewish/Kosher	[{'date': 2014-11-10 00:00:00, 'grade':	Seuda Foods	40360045

Exemples $unwind

```
# Séparation des 5 premiers restaurants sur la base des évaluations (grades).
# Chaque ligne résultat correspond maintenant a une évaluation pour un restaurant
print(pandas.DataFrame(db.restaurants.aggregate(
    [    { "$limit": 5 },
         { "$unwind": "$grades" }
    ])))
```

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2014-03-03 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
1	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2013-09-11 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
2	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2013-01-24 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
3	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2011-11-23 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
4	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2011-03-10 00:00:00, 'grade': 'B', 's...	Morris Park Bake Shop	30075445
5	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	{'date': 2014-12-30 00:00:00, 'grade': 'A', 's...	Wendy'S	30112340
		{'building': '469', 'coord':			{'date': 2014-07-01		

Exemples $unwind

```
# Idem avec sélection des évaluations à « B »
print(pandas.DataFrame(db.restaurants.aggregate(
    [    { "$limit": 5 },
         { "$unwind": "$grades" },
         { "$match": { "grades.grade": "B" }}
    ])))
```

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2011-03-10 00:00:00, 'grade': 'B', 's...	Morris Park Bake Shop	30075445
1	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	{'date': 2014-07-01 00:00:00, 'grade': 'B', 's...	Wendy'S	30112340
2	60006d6aa7aafd5a6d45ca9e	{'building': '97-22', 'coord': [-73.8601152, 4...	Queens	Jewish/Kosher	{'date': 2011-12-15 00:00:00, 'grade': 'B', 's...	Tov Kosher Kitchen	40356068

Exemples $unwind

```
# Si on inverse les opérations, résultat différend
print(pandas.DataFrame(db.restaurants.aggregate(
    [    { "$limit": 5 },
         { "$match": { "grades.grade": "B" }},
         { "$unwind": "$grades" }
    ])))
```

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2014-03-03 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
1	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2013-09-11 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
2	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2013-01-24 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
3	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2011-11-23 00:00:00, 'grade': 'A', 's...	Morris Park Bake Shop	30075445
4	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	{'date': 2011-03-10 00:00:00, 'grade': 'B', 's...	Morris Park Bake Shop	30075445
5	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	{'date': 2014-12-30 00:00:00, 'grade': 'A', 's...	Wendy'S	30112340
6	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	{'date': 2014-07-01 00:00:00, 'grade': 'B', 's...	Wendy'S	30112340
7	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	{'date': 2013-04-30 00:00:00, 'grade': 'A', 's...	Wendy'S	30112340
8	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	{'date': 2012-05-08 00:00:00, 'grade': 'A', 's...	Wendy'S	30112340
9	60006d6aa7aafd5a6d45ca9e	{'building': '97-22',	Queens	Jewish/Kosher	{'date': 2014-11-24	Tov Kosher	40356068

L'opération `$project` : redéfinition des documents

- **`{"champ" : 1}`** : conservation du champ (0 = suppression – idem que pour `find`)
- **`{"champ":{"$opérateur":expression}}`** : permet de définir un nouveau champ
- **`{"nouveauChamp" : "ancienChamp"}}`** : renommage

Quelques opérateurs utiles pour la projection :

- `$arrayElemAt` : élément d'un tableau
- `$first` et `$last` : premier ou dernier élément du tableau
- `$size` : taille d'un tableau
- `$substr` : sous-chaîne de caractères
- `$cond` : permet de faire une condition (genre de *if then else*)
- …

Plus d'informations : https://www.mongodb.com/docs/manual/reference/operator/aggregation/

Exemples $project

```
# nom et quartier des 10 premiers restaurants.
# L'ordre des champs projetés est celui de la base, pas celui de la requête
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$project": { "name": 1, "borough": 1 } }
    ])))
```

	_id	borough	name
0	60006d6aa7aafd5a6d45ca9a	Bronx	Morris Park Bake Shop
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	Wendy'S
2	60006d6aa7aafd5a6d45ca9c	Manhattan	Dj Reynolds Pub And Restaurant
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	Riviera Caterer
4	60006d6aa7aafd5a6d45ca9e	Queens	Tov Kosher Kitchen
5	60006d6aa7aafd5a6d45ca9f	Queens	Brunos On The Boulevard
6	60006d6aa7aafd5a6d45caa0	Staten Island	Kosher Island
7	60006d6aa7aafd5a6d45caa1	Brooklyn	Wilken'S Fine Food
8	60006d6aa7aafd5a6d45caa2	Brooklyn	Regina Caterers
9	60006d6aa7aafd5a6d45caa3	Brooklyn	Taste The Tropics Ice Cream

Exemples $project

```
# Eviction de certains champs
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$project": { "address": 0, "grades": 0 } }
    ])))
```

	_id	borough	cuisine	name	restaurant_id
0	60006d6aa7aafd5a6d45ca9a	Bronx	Bakery	Morris Park Bake Shop	30075445
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	Hamburgers	Wendy'S	30112340
2	60006d6aa7aafd5a6d45ca9c	Manhattan	Irish	Dj Reynolds Pub And Restaurant	30191841
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	American	Riviera Caterer	40356018
4	60006d6aa7aafd5a6d45ca9e	Queens	Jewish/Kosher	Tov Kosher Kitchen	40356068
5	60006d6aa7aafd5a6d45ca9f	Queens	American	Brunos On The Boulevard	40356151
6	60006d6aa7aafd5a6d45caa0	Staten Island	Jewish/Kosher	Kosher Island	40356442
7	60006d6aa7aafd5a6d45caa1	Brooklyn	Delicatessen	Wilken'S Fine Food	40356483
8	60006d6aa7aafd5a6d45caa2	Brooklyn	American	Regina Caterers	40356649
9	60006d6aa7aafd5a6d45caa3	Brooklyn	Ice Cream, Gelato, Yogurt, Ices	Taste The Tropics Ice Cream	40356731

Exemples $project

```
# Renommage de champs
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$project": { "name": 1, "borough": 1 , "street": "$address.street"} }
    ])))
```

	_id	borough	name	street
0	60006d6aa7aafd5a6d45ca9a	Bronx	Morris Park Bake Shop	Morris Park Ave
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	Wendy'S	Flatbush Avenue
2	60006d6aa7aafd5a6d45ca9c	Manhattan	Dj Reynolds Pub And Restaurant	West 57 Street
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	Riviera Caterer	Stillwell Avenue
4	60006d6aa7aafd5a6d45ca9e	Queens	Tov Kosher Kitchen	63 Road
5	60006d6aa7aafd5a6d45ca9f	Queens	Brunos On The Boulevard	Astoria Boulevard
6	60006d6aa7aafd5a6d45caa0	Staten Island	Kosher Island	Victory Boulevard
7	60006d6aa7aafd5a6d45caa1	Brooklyn	Wilken'S Fine Food	Avenue U
8	60006d6aa7aafd5a6d45caa2	Brooklyn	Regina Caterers	11 Avenue
9	60006d6aa7aafd5a6d45caa3	Brooklyn	Taste The Tropics Ice Cream	Nostrand Avenue

Exemples $project

```
# Nb visites (évaluations) par restaurant (taille du tableau grades)
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$project": {"name":1,"borough":1,"nb_grades":{"$size":"$grades"}}}
    ])))
```

	_id	borough	name	nb_grades
0	60006d6aa7aafd5a6d45ca9a	Bronx	Morris Park Bake Shop	5
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	Wendy'S	4
2	60006d6aa7aafd5a6d45ca9c	Manhattan	Dj Reynolds Pub And Restaurant	4
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	Riviera Caterer	4
4	60006d6aa7aafd5a6d45ca9e	Queens	Tov Kosher Kitchen	4
5	60006d6aa7aafd5a6d45ca9f	Queens	Brunos On The Boulevard	4
6	60006d6aa7aafd5a6d45caa0	Staten Island	Kosher Island	4
7	60006d6aa7aafd5a6d45caa1	Brooklyn	Wilken'S Fine Food	6
8	60006d6aa7aafd5a6d45caa2	Brooklyn	Regina Caterers	5
9	60006d6aa7aafd5a6d45caa3	Brooklyn	Taste The Tropics Ice Cream	4

Exemples $project

```
# Nb visites (évaluations) par restaurant, tri décroissant puis affichage des 10 premiers
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$project":{"name":1,"borough":1,"nb_grades":{"$size":"$grades" }}},
        { "$sort": { "nb_grades": -1 }},
        { "$limit": 10 }
    ])))
```

	_id	borough	name	nb_grades
0	60006d6ca7aafd5a6d45f30a	Brooklyn	Silver Krust West Indian Restaurant	9
1	60006d6ba7aafd5a6d45e7c6	Brooklyn	Lai Lai Gourmet	9
2	60006d6da7aafd5a6d46011f	Manhattan	Pure Food	9
3	60006d6ca7aafd5a6d45f9cf	Manhattan	Breeze Thai-French Kitchen	9
4	60006d6ba7aafd5a6d45e35d	Manhattan	Benton	9
5	60006d6ca7aafd5a6d45e9fd	Manhattan	Nomado 33	9
6	60006d6ba7aafd5a6d45e4ce	Brooklyn	Noodle Station	9
7	60006d6ba7aafd5a6d45e332	Manhattan	S'Mac	9
8	60006d6aa7aafd5a6d45ceeb	Manhattan	World Cup Cafe	8
9	60006d6aa7aafd5a6d45cf32	Brooklyn	Fifth Ave Cafe /Diner	8

Exemples $project

```
# Affichage du 1er élément des grades (indice 0)
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$project":{"name":1,"borough":1,"grade":{"$arrayElemAt":["$grades",0]}}}
    ])))
```

	_id	borough	name	grade
0	60006d6aa7aafd5a6d45ca9a	Bronx	Morris Park Bake Shop	{'date': 2014-03-03 00:00:00, 'grade': 'A', 's...
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	Wendy'S	{'date': 2014-12-30 00:00:00, 'grade': 'A', 's...
2	60006d6aa7aafd5a6d45ca9c	Manhattan	Dj Reynolds Pub And Restaurant	{'date': 2014-09-06 00:00:00, 'grade': 'A', 's...
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	Riviera Caterer	{'date': 2014-06-10 00:00:00, 'grade': 'A', 's...
4	60006d6aa7aafd5a6d45ca9e	Queens	Tov Kosher Kitchen	{'date': 2014-11-24 00:00:00, 'grade': 'Z', 's...
5	60006d6aa7aafd5a6d45ca9f	Queens	Brunos On The Boulevard	{'date': 2014-11-15 00:00:00, 'grade': 'Z', 's...
6	60006d6aa7aafd5a6d45caa0	Staten Island	Kosher Island	{'date': 2014-10-06 00:00:00, 'grade': 'A', 's...
7	60006d6aa7aafd5a6d45caa1	Brooklyn	Wilken'S Fine Food	{'date': 2014-05-29 00:00:00, 'grade': 'A', 's...
8	60006d6aa7aafd5a6d45caa2	Brooklyn	Regina Caterers	{'date': 2014-07-18 00:00:00, 'grade': 'A', 's...
9	60006d6aa7aafd5a6d45caa3	Brooklyn	Taste The Tropics Ice Cream	{'date': 2014-07-14 00:00:00, 'grade': 'A', 's...

```
# Idem avec {"$project":{"name":1,"borough":1,"grade":{"$first": "$grades" } } }
# $last pour le dernier élément
```

Exemples $project

```
# Mettre en majuscule
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$project": { "nom": { "$toUpper": "$name" }, "borough": 1 } }
    ])))
```

	_id	borough	nom
0	60006d6aa7aafd5a6d45ca9a	Bronx	MORRIS PARK BAKE SHOP
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	WENDY'S
2	60006d6aa7aafd5a6d45ca9c	Manhattan	DJ REYNOLDS PUB AND RESTAURANT
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	RIVIERA CATERER
4	60006d6aa7aafd5a6d45ca9e	Queens	TOV KOSHER KITCHEN
5	60006d6aa7aafd5a6d45ca9f	Queens	BRUNOS ON THE BOULEVARD
6	60006d6aa7aafd5a6d45caa0	Staten Island	KOSHER ISLAND
7	60006d6aa7aafd5a6d45caa1	Brooklyn	WILKEN'S FINE FOOD
8	60006d6aa7aafd5a6d45caa2	Brooklyn	REGINA CATERERS
9	60006d6aa7aafd5a6d45caa3	Brooklyn	TASTE THE TROPICS ICE CREAM

Exemples $project

```
# On ajoute un nouveau champ
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$addFields": { "nb_grades": { "$size": "$grades" } } }
    ])))
```

	_id	address	borough	cuisine	grades	name	restaurant_id	nb_grades
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	[{'date': 2014-03-03 00:00:00, 'grade': 'A', '...	Morris Park Bake Shop	30075445	5
1	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	[{'date': 2014-12-30 00:00:00, 'grade': 'A', '...	Wendy'S	30112340	4
2	60006d6aa7aafd5a6d45ca9c	{'building': '351', 'coord': [-73.985135599999...	Manhattan	Irish	[{'date': 2014-09-06 00:00:00, 'grade': 'A', '...	Dj Reynolds Pub And Restaurant	30191841	4
3	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018	4
4	60006d6aa7aafd5a6d45ca9e	{'building': '97-22', 'coord': [-73.8601152, 4...	Queens	Jewish/Kosher	[{'date': 2014-11-24 00:00:00, 'grade': 'Z', '...	Tov Kosher Kitchen	40356068	4
5	60006d6aa7aafd5a6d45ca9f	{'building': '8825', 'coord': [-73.8803827,	Queens	American	[{'date': 2014-11-15	Brunos On The	40356151	4

Exemples $project

```
# Extraction des 3 premières lettres des quartiers (sous-chaîne)
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$project": {
            "nom": { "$toUpper": "$name" },
            "quartier": { "$substr": [ "$borough", 0, 3 ] }
        }}
    ])))
```

	_id	nom	quartier
0	60006d6aa7aafd5a6d45ca9a	MORRIS PARK BAKE SHOP	Bro
1	60006d6aa7aafd5a6d45ca9b	WENDY'S	Bro
2	60006d6aa7aafd5a6d45ca9c	DJ REYNOLDS PUB AND RESTAURANT	Man
3	60006d6aa7aafd5a6d45ca9d	RIVIERA CATERER	Bro
4	60006d6aa7aafd5a6d45ca9e	TOV KOSHER KITCHEN	Que
5	60006d6aa7aafd5a6d45ca9f	BRUNOS ON THE BOULEVARD	Que
6	60006d6aa7aafd5a6d45caa0	KOSHER ISLAND	Sta
7	60006d6aa7aafd5a6d45caa1	WILKEN'S FINE FOOD	Bro
8	60006d6aa7aafd5a6d45caa2	REGINA CATERERS	Bro
9	60006d6aa7aafd5a6d45caa3	TASTE THE TROPICS ICE CREAM	Bro

Exemples $project

```
# Idem mais BRX pour le BRONX sinon on garde les 3 premières lettres du quartier
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$addFields": {"quartier":{"$toUpper":{"$substr":["$borough",0,3]}}}},
        { "$project": {
            "nom": { "$toUpper": "$name" },
            "quartier": { "$cond": {
                "if": { "$eq": ["$borough", "Bronx"]},
                "then": "BRX",
                "else": "$quartier"
            } },
            "borough": 1
        }}
    ])))
```

	_id	borough	nom	quartier
0	60006d6aa7aafd5a6d45ca9a	Bronx	MORRIS PARK BAKE SHOP	BRX
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	WENDY'S	BRO
2	60006d6aa7aafd5a6d45ca9c	Manhattan	DJ REYNOLDS PUB AND RESTAURANT	MAN
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	RIVIERA CATERER	BRO
4	60006d6aa7aafd5a6d45ca9e	Queens	TOV KOSHER KITCHEN	QUE
5	60006d6aa7aafd5a6d45ca9f	Queens	BRUNOS ON THE BOULEVARD	QUE
6	60006d6aa7aafd5a6d45caa0	Staten Island	KOSHER ISLAND	STA
7	60006d6aa7aafd5a6d45caa1	Brooklyn	WILKEN'S FINE FOOD	BRO
8	60006d6aa7aafd5a6d45caa2	Brooklyn	REGINA CATERERS	BRO
9	60006d6aa7aafd5a6d45caa3	Brooklyn	TASTE THE TROPICS ICE CREAM	BRO

L'opération `$group` : calcul d'agrégats

- **`_id :`** déclaration du critère de regroupement
  - Chaine de caractères : pas de regroupement (tous les documents)
  - **`$champ`** : regroupement selon ce champ
  - **`{"$a1":"champ1",…}`** **`:`** regroupement multiple avec modification possible des valeurs

- Calcul d'agrégat à faire :
  - **`$sum`** : somme (à `1` pour faire un décompte(count) sinon sur un champ spécifique)
  - **`$avg, $min, $max`**
  - **`$addToSet`** : regroupement des valeurs distinctes d'un champ dans un tableau

Exemples $group

```
# Nombre de restaurants par quartier (trié)
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        {"$group": {"_id": "$borough", "NbRestos": {"$sum": 1}}},
        {"$sort": { "NbRestos": -1}}
    ])))
```

	_id	NbRestos
0	Manhattan	10259
1	Brooklyn	6086
2	Queens	5656
3	Bronx	2338
4	Staten Island	969
5	Missing	51

Cette requête donne le même résultat :

```
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        {"$sortByCount": "$borough"}
    ])))
```

Exemples $group

```
# Note moyenne des restaurants du Queens
print(pandas.DataFrame( db.restaurants.aggregate(
    [
        { "$match": { "borough": "Queens" }},
        { "$unwind": "$grades" },
        { "$group": { "_id": "null", "score": { "$avg": "$grades.score" }}}
    ])))
```

	_id	score
0	null	11.634865

Exemples $group

```
# Note moyenne des restaurants par quartier
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$unwind": "$grades" },
        { "$group": { "_id": "$borough", "score": { "$avg": "$grades.score" }}},
        { "$sort": { "score": -1 }}
    ])))
```

	_id	score
0	Queens	11.634865
1	Brooklyn	11.447976
2	Manhattan	11.418151
3	Staten Island	11.370958
4	Bronx	11.036186
5	Missing	9.632911

Exemples $group

```
# regroupement par quartier et par rue (en ne prenant que la première évaluation - qui est la dernière en
# date a priori), pour afficher les 10 rues où on mange le plus sainement
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$project": {
            "borough": 1, "street": "$address.street",
            "eval": { "$arrayElemAt": [ "$grades", 0 ]}
        } },
        { "$match": { "eval": { "$exists": True } } },    # Supprime les restos sans eval
        { "$group": {
            "_id": { "quartier": "$borough", "rue": "$street" },
            "score": { "$avg": "$eval.score" }
        }},
        { "$sort": { "score": 1 }},
        { "$limit": 10 }
    ])))
```

	_id	score
0	{'quartier': 'Manhattan', 'rue': '106 Street &...	-1.0
1	{'quartier': 'Brooklyn', 'rue': 'Shore Pkwy So...	0.0
2	{'quartier': 'Manhattan', 'rue': 'Oliver St'}	0.0
3	{'quartier': 'Brooklyn', 'rue': '78Th St'}	0.0
4	{'quartier': 'Brooklyn', 'rue': 'Stockholm St'}	0.0

Exemples $group

```
# Pour comprendre la différence entre $addToSet et $push, on les applique sur les grades
# obtenus pour les 10 premiers restaurants
# $addToSet : valeurs distinctes
# $push : toutes les valeurs présentes
print(pandas.DataFrame(db.restaurants.aggregate(
    [
        { "$limit": 10 },
        { "$unwind": "$grades" },
        { "$group": {
            "_id": "$name",
            "avec_addToSet": { "$addToSet": "$grades.grade" },
            "avec_push": { "$push": "$grades.grade" }
        }}
    ])))
```

	_id	avec_addToSet	avec_push
0	Tov Kosher Kitchen	[B, Z, A]	[Z, A, A, B]
1	Riviera Caterer	[A]	[A, A, A, A]
2	Kosher Island	[A]	[A, A, A, A]
3	Wendy'S	[B, A]	[A, B, A, A]
4	Morris Park Bake Shop	[B, A]	[A, A, A, A, B]
5	Wilken'S Fine Food	[A]	[A, A, A, A, A, A]
6	Regina Caterers	[A]	[A, A, A, A, A]
7	Dj Reynolds Pub And Restaurant	[A]	[A, A, A, A]
8	Brunos On The Boulevard	[Z, A]	[Z, A, A, A]
9	Taste The Tropics Ice Cream	[A]	[A, A, A, A]

Chap. 3 – Python et MongoDB

Itération

Il est possible de définir un curseur qui va itérer sur la liste de résultats (stockée sur le serveur). Cela permet de récupérer les documents par paquets, ce qui est judicieux en cas de gros volume *(pour éviter de congestionner un réseau par exemple).*

```
#les restaurants qui ont eu plus de 49 en score
print(pandas.DataFrame(db.restaurants.find(
    {"borough": "Queens", "grades.score": { "$gte":  50}},
    {"_id": 0, "name": 1, "address.street": 1},
    batch_size = 10))
```

	address	name
0	{'street': 'Horace Harding Boulevard'}	Richer'S Bakery
1	{'street': 'Bell Boulevard'}	Tequilla Sunrise
2	{'street': 'Rockaway Beach Boulevard'}	Rockaway Beach Inn
3	{'street': 'Broadway'}	Alfonso'S Bar
4	{'street': 'Woodhaven Boulevard'}	Pio Pio
...	...	...
78	{'street': '30 Avenue'}	Queens Comfort Restaurant
79	{'street': 'Union Turnpike'}	Koyla
80	{'street': '37 Road'}	Jackson Heights Food Court
81	{'street': 'Union Street'}	K & D Internet Inc
82	{'street': 'Roosevelt Ave'}	Tacos Al Suadero

Mais l'opération est totalement transparente dans python, puisque lorsque nous appelons le curseur, nous récupérons tous les documents.

Chap. 3 – Python et MongoDB

Variables complexes dans un DataFrame : Une fois importées dans un DataFrame, les champs complexes (comme `address` et `grades`) sont des variables d'un type un peu particulier

	_id	address	borough	cuisine	grades	name	restaurant_id
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	[{'date': 2014-03-03 00:00:00, 'grade': 'A', '...	Morris Park Bake Shop	30075445
1	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	[{'date': 2014-12-30 00:00:00, 'grade': 'A', '...	Wendy'S	30112340
2	60006d6aa7aafd5a6d45ca9c	{'building': '351', 'coord': [-73.985135599999...	Manhattan	Irish	[{'date': 2014-09-06 00:00:00, 'grade': 'A', '...	Dj Reynolds Pub And Restaurant	30191841
3	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018
4	60006d6aa7aafd5a6d45ca9e	{'building': '97-22', 'coord': [-73.8601152, 4...	Queens	Jewish/Kosher	[{'date': 2014-11-24 00:00:00, 'grade': 'Z', '...	Tov Kosher Kitchen	40356068
5	60006d6aa7aafd5a6d45ca9f	{'building': '8825', 'coord': [-73.8803827, 40...	Queens	American	[{'date': 2014-11-15 00:00:00, 'grade': 'Z', '...	Brunos On The Boulevard	40356151
					[{'date':		

Focus sur le champ `address` : c'est une liste de **dictionnaires**, ayant chacun plusieurs champs (ici tous les mêmes).

```
0    {'building': '1007', 'coord': [-73.856077, 40....
1    {'building': '469', 'coord': [-73.961704, 40.6...
2    {'building': '351', 'coord': [-73.985135599999...
3    {'building': '2780', 'coord': [-73.98241999999...
4    {'building': '97-22', 'coord': [-73.8601152, 4...
5    {'building': '8825', 'coord': [-73.8803827, 40...
6    {'building': '2206', 'coord': [-74.1377286, 40...
7    {'building': '7114', 'coord': [-73.9068506, 40...
8    {'building': '6409', 'coord': [-74.00528899999...
9    {'building': '1839', 'coord': [-73.9482609, 40...
Name: address, dtype: object
```

Chap. 3 – Python et MongoDB

Nouveau champ dans la DataFrame `df` : *list comprehension* pour créer une concaténation du bâtiment et de la rue :

```
df.assign(info = [e["building"] + ", " + e["street"] for e in df.address])
```

	_id	address	borough	cuisine	grades	name	restaurant_id	info
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	[{'date': 2014-03-03 00:00:00, 'grade': 'A', '...	Morris Park Bake Shop	30075445	1007, Morris Park Ave
1	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	[{'date': 2014-12-30 00:00:00, 'grade': 'A', '...	Wendy'S	30112340	469, Flatbush Avenue
2	60006d6aa7aafd5a6d45ca9c	{'building': '351', 'coord': [-73.985135599999...	Manhattan	Irish	[{'date': 2014-09-06 00:00:00, 'grade': 'A', '...	Dj Reynolds Pub And Restaurant	30191841	351, West 57 Street
3	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018	2780, Stillwell Avenue
4	60006d6aa7aafd5a6d45ca9e	{'building': '97-22', 'coord': [-73.8601152, 4...	Queens	Jewish/Kosher	[{'date': 2014-11-24 00:00:00, 'grade': 'Z', '...	Tov Kosher Kitchen	40356068	97-22, 63 Road
5	60006d6aa7aafd5a6d45ca9f	{'building': '8825', 'coord': [-73.8803827, 40...	Queens	American	[{'date': 2014-11-15 00:00:00, 'grade': 'Z', '...	Brunos On The Boulevard	40356151	8825, Astoria Boulevard
6	60006d6aa7aafd5a6d45caa0	{'building': '2206', 'coord': [-74.1377286, 40...	Staten Island	Jewish/Kosher	[{'date': 2014-10-06 00:00:00, 'grade': 'A', '...	Kosher Island	40356442	2206, Victory Boulevard

Chap. 3 – Python et MongoDB

Transformation de la liste `address` en DataFrame

```
pandas.DataFrame([e for e in df.address])
```

	building	coord	street	zipcode
0	1007	[-73.856077, 40.848447]	Morris Park Ave	10462
1	469	[-73.961704, 40.662942]	Flatbush Avenue	11225
2	351	[-73.98513559999999, 40.7676919]	West 57 Street	10019
3	2780	[-73.98241999999999, 40.579505]	Stillwell Avenue	11224
4	97-22	[-73.8601152, 40.7311739]	63 Road	11374
5	8825	[-73.8803827, 40.7643124]	Astoria Boulevard	11369
6	2206	[-74.1377286, 40.6119572]	Victory Boulevard	10314
7	7114	[-73.9068506, 40.6199034]	Avenue U	11234
8	6409	[-74.00528899999999, 40.628886]	11 Avenue	11219
9	1839	[-73.9482609, 40.6408271]	Nostrand Avenue	11226

Chap. 3 – Python et MongoDB

Idem dans le DataFrame d'origine

```
pandas.concat([df.drop("address", axis = 1), pandas.DataFrame([e for e in df.address])], axis = 1)
```

	_id	borough	cuisine	grades	name	restaurant_id	building	coord	street
0	60006d6aa7aafd5a6d45ca9a	Bronx	Bakery	[{'date': 2014-03-03 00:00:00, 'grade': 'A', '...	Morris Park Bake Shop	30075445	1007	[-73.856077, 40.848447]	Morris Park Ave
1	60006d6aa7aafd5a6d45ca9b	Brooklyn	Hamburgers	[{'date': 2014-12-30 00:00:00, 'grade': 'A', '...	Wendy'S	30112340	469	[-73.961704, 40.662942]	Flatbush Avenue
2	60006d6aa7aafd5a6d45ca9c	Manhattan	Irish	[{'date': 2014-09-06 00:00:00, 'grade': 'A', '...	Dj Reynolds Pub And Restaurant	30191841	351	[-73.98513559999999, 40.7676919]	West 57 Street
3	60006d6aa7aafd5a6d45ca9d	Brooklyn	American	[{'date': 2014-06-10 00:00:00, 'grade': 'A', '...	Riviera Caterer	40356018	2780	[-73.98241999999999, 40.579505]	Stillwell Avenue
4	60006d6aa7aafd5a6d45ca9e	Queens	Jewish/Kosher	[{'date': 2014-11-24 00:00:00, 'grade': 'Z', '...	Tov Kosher Kitchen	40356068	97-22	[-73.8601152, 40.7311739]	63 Road
5	60006d6aa7aafd5a6d45ca9f	Queens	American	[{'date': 2014-11-15 00:00:00, 'grade': 'Z', '...	Brunos On The Boulevard	40356151	8825	[-73.8803827, 40.7643124]	Astoria Boulevard
6	60006d6aa7aafd5a6d45caa0	Staten	Jewish/Kosher	[{'date': 2014-10-06 00:00:00	Kosher	40356442	2206	[-74.1377286,	Victory

Variable Tableau : Le champ `grades` est une liste de tableaux, ayant chacun potentiellement plusieurs valeurs (des dictionnaires)

```
df.grades
```

```
0    [{'date': 2014-03-03 00:00:00, 'grade': 'A', '...
1    [{'date': 2014-12-30 00:00:00, 'grade': 'A', '...
2    [{'date': 2014-09-06 00:00:00, 'grade': 'A', '...
3    [{'date': 2014-06-10 00:00:00, 'grade': 'A', '...
4    [{'date': 2014-11-24 00:00:00, 'grade': 'Z', '...
5    [{'date': 2014-11-15 00:00:00, 'grade': 'Z', '...
6    [{'date': 2014-10-06 00:00:00, 'grade': 'A', '...
7    [{'date': 2014-05-29 00:00:00, 'grade': 'A', '...
8    [{'date': 2014-07-18 00:00:00, 'grade': 'A', '...
9    [{'date': 2014-07-14 00:00:00, 'grade': 'A', '...
Name: grades, dtype: object
```

Récupération d'un élément du tableau

```
df.assign(derniere = [e[0] for e in df.grades], premiere = [e[-1] for e in df.grades]).drop("gra
```

	_id	address	borough	cuisine	name	restaurant_id	derniere	premiere
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	Morris Park Bake Shop	30075445	{'date': 2014-03-03 00:00:00, 'grade': 'A', 's...	{'date': 2011-03-10 00:00:00, 'grade': 'B', 's...
1	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	Wendy'S	30112340	{'date': 2014-12-30 00:00:00, 'grade': 'A', 's...	{'date': 2012-05-08 00:00:00, 'grade': 'A', 's...
2	60006d6aa7aafd5a6d45ca9c	{'building': '351', 'coord': [-73.985135599999...	Manhattan	Irish	Dj Reynolds Pub And Restaurant	30191841	{'date': 2014-09-06 00:00:00, 'grade': 'A', 's...	{'date': 2011-12-29 00:00:00, 'grade': 'A', 's...
3	60006d6aa7aafd5a6d45ca9d	{'building': '2780', 'coord': [-73.98241999999...	Brooklyn	American	Riviera Caterer	40356018	{'date': 2014-06-10 00:00:00, 'grade': 'A', 's...	{'date': 2011-10-12 00:00:00, 'grade': 'A', 's...
4	60006d6aa7aafd5a6d45ca9e	{'building': '97-22', 'coord': [-73.8601152, 4...	Queens	Jewish/Kosher	Tov Kosher Kitchen	40356068	{'date': 2014-11-24 00:00:00, 'grade': 'Z', 's...	{'date': 2011-12-15 00:00:00, 'grade': 'B', 's...
5	60006d6aa7aafd5a6d45ca9f	{'building': '8825', 'coord': [-73.8803827, 40...	Queens	American	Brunos On The Boulevard	40356151	{'date': 2014-11-15 00:00:00, 'grade': 'Z', 's...	{'date': 2012-02-10 00:00:00, 'grade': 'A', 's...

Chap. 3 – Python et MongoDB

Transformation de la liste de tableaux en un seul DataFrame

`zip()` permet d'itérer sur plusieurs tableaux en même temps

`concat()` permet de concaténer les tableaux entre eux

```
dfgrades = pandas.concat([pandas.DataFrame(g).assign(_id = i) for (i, g) in zip(df._id, df.grades)])
dfgrades
```

	date	grade	score	_id
0	2014-03-03	A	2	60006d6aa7aafd5a6d45ca9a
1	2013-09-11	A	6	60006d6aa7aafd5a6d45ca9a
2	2013-01-24	A	10	60006d6aa7aafd5a6d45ca9a
3	2011-11-23	A	9	60006d6aa7aafd5a6d45ca9a
4	2011-03-10	B	14	60006d6aa7aafd5a6d45ca9a
0	2014-12-30	A	8	60006d6aa7aafd5a6d45ca9b
1	2014-07-01	B	23	60006d6aa7aafd5a6d45ca9b
2	2013-04-30	A	12	60006d6aa7aafd5a6d45ca9b
3	2012-05-08	A	12	60006d6aa7aafd5a6d45ca9b
0	2014-09-06	A	2	60006d6aa7aafd5a6d45ca9c
1	2013-07-22	A	11	60006d6aa7aafd5a6d45ca9c
2	2012-07-31	A	12	60006d6aa7aafd5a6d45ca9c
3	2011-12-29	A	12	60006d6aa7aafd5a6d45ca9c
0	2014-06-10	A	5	60006d6aa7aafd5a6d45ca9d
1	2013-06-05	A	7	60006d6aa7aafd5a6d45ca9d
2	2012-04-13	A	12	60006d6aa7aafd5a6d45ca9d
3	2011-10-12	A	12	60006d6aa7aafd5a6d45ca9d
0	2014-11-24	Z	20	60006d6aa7aafd5a6d45ca9e
1	2013-01-17	A	13	60006d6aa7aafd5a6d45ca9e
2	2012-08-02	A	13	60006d6aa7aafd5a6d45ca9e

Jointure entre deux DataFrames avec Merge

```
pandas.merge(df.drop("grades", axis = 1), dfgrades.reset_index())
```

	_id	address	borough	cuisine	name	restaurant_id	index	date	grade	score
0	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	Morris Park Bake Shop	30075445	0	2014-03-03	A	2
1	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	Morris Park Bake Shop	30075445	1	2013-09-11	A	6
2	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	Morris Park Bake Shop	30075445	2	2013-01-24	A	10
3	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	Morris Park Bake Shop	30075445	3	2011-11-23	A	9
4	60006d6aa7aafd5a6d45ca9a	{'building': '1007', 'coord': [-73.856077, 40....	Bronx	Bakery	Morris Park Bake Shop	30075445	4	2011-03-10	B	14
5	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	Wendy'S	30112340	0	2014-12-30	A	8
6	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	Wendy'S	30112340	1	2014-07-01	B	23
7	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	Wendy'S	30112340	2	2013-04-30	A	12
8	60006d6aa7aafd5a6d45ca9b	{'building': '469', 'coord': [-73.961704, 40.6...	Brooklyn	Hamburgers	Wendy'S	30112340	3	2012-05-08	A	12
		{'building': '351',			Dj					

Exercice 1 : Restaurants NewYorkais

1. Donner les styles de cuisine présent dans la collection
2. Donner tous les grades possibles dans la base
3. Compter le nombre de restaurants proposant de la cuisine française (“French”)
4. Compter le nombre de restaurants situé sur la rue “Central Avenue”
5. Compter le nombre de restaurants ayant eu une note supérieure à 50
6. Lister tous les restaurants, en n’affichant que le nom, l’immeuble et la rue
7. Lister tous les restaurants nommés “Burger King” (adresse rue et quartier uniquement) trié par quartier puis rue décroissante
8. Lister les restaurants situés sur les rues “Union Street” ou “Union Square”
9. Lister les restaurants situés au-dessus de la latitude 40.90
10. Lister les restaurants ayant eu un score de 0 et un grade “A”
11. Lister les restaurants (nom et rue uniquement) situés sur une rue ayant le terme “Union” dans le nom
12. Lister les restaurants ayant eu une visite le 1er février 2014
13. Lister les restaurants situés entre les longitudes -74.2 et -74.1 et les latitudes 40.1 et 40.2

Exercice 2 Restaurants new-yorkais (suite)

1. Quelles sont les 10 plus grandes chaines de restaurants (nom identique avec le plus de restaurants) ?
2. Donner le Top 5 et le Flop 5 des types de cuisine, en terme de nombre de restaurants (2 requêtes)
3. Quelles sont les 10 rues avec le plus de restaurants ?
4. Quelles sont les rues situées sur exactement 2 quartiers ?
   Essayez d'ajouter le nom des quartiers de chaque rue (cf addToSet voir diapo 49)
5. Lister par quartier le nombre de restaurants et le score moyen ? (Attention à bien découper le tableau `grades`)
6. Donner les dates de début et de fin de la période des évaluations de ce jeu de données
7. Quels sont les 10 restaurants (nom, quartier, adresse et score) avec le plus petit score moyen ? (ôter les None dans score)
8. Quels sont les restaurants (nom, quartier et adresse rue) avec uniquement des grades "A" ?
   - *restriction à ceux qui ont A, découpage, suppression des autres grades que "A" et affichage des infos*
   - *on peut envisager d'autres choses (découpage, addToSet, et restriction à ceux pour lequel le tableau créé = ["A"] - par exemple)*
9. Compter le nombre d'évaluations par jour de la semaine (jour en numérique + simple) –trié sur le nombre d'évaluation décr
   petite recherche sur l'extraction du jour de la semaine à partir d'une date à faire
10. Donner les 3 types de cuisine les plus présents par quartier
    piste :
    1. *double regroupement à prévoir*
    2. *tri à prévoir*
    3. *regroupement avec push*
    4. *slice pour prendre une partie d'un tableau*

Exercice 3 : AirBnB

explication des données : https://www.mongodb.com/docs/atlas/sample-data/sample-airbnb/

1. Donner le nombre de logements
2. Lister les différentes types de logements possibles cf (room_type)
3. Lister les différents équipements possibles cf (amenities)
4. Donner le nombre de logements de type “Entire home/apt”
5. Donner le nombre de logements proposant la “TV” et le “Wifi (cf amenities)
6. Donner le nombre de logements n’ayant eu aucun avis (champs number_of_reviews et reviews (tableau des avis))
7. Lister les informations du logement “10545725” (sans _id)
8. Lister le nom, la rue et le pays des logements dont le prix est supérieur à 10 000.
9. Donner le nombre de logements par type
10. Donner le nombre de logements par pays
11. Calculer pour chaque type de logements (room_type) la moyenne des prix (price) avec arrondi à 2 décimales et un graphique associé.
12. Pour chaque logement, quel est le nombre d’avis ?
13. Compter le nombre de logements pour chaque équipement possible
14. On souhaite connaître les 10 utilisateurs (id et nom) ayant fait le plus de commentaires