*Nom de la team : RSA*

*Sprint n°1 - Projet N°6 Criminalité à Chicago*

*Scrum Master : Ryme Youb*

*Teammates : Adrien Golebiewski, Sacha Cymermann*

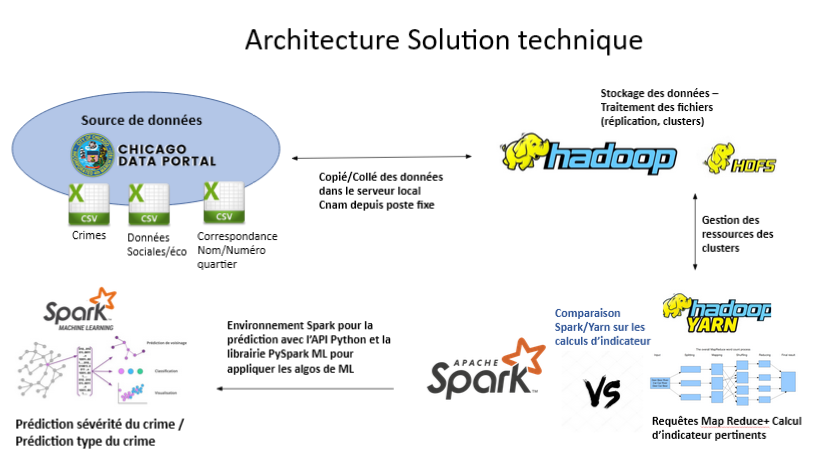
*Début : 19/11/2021*

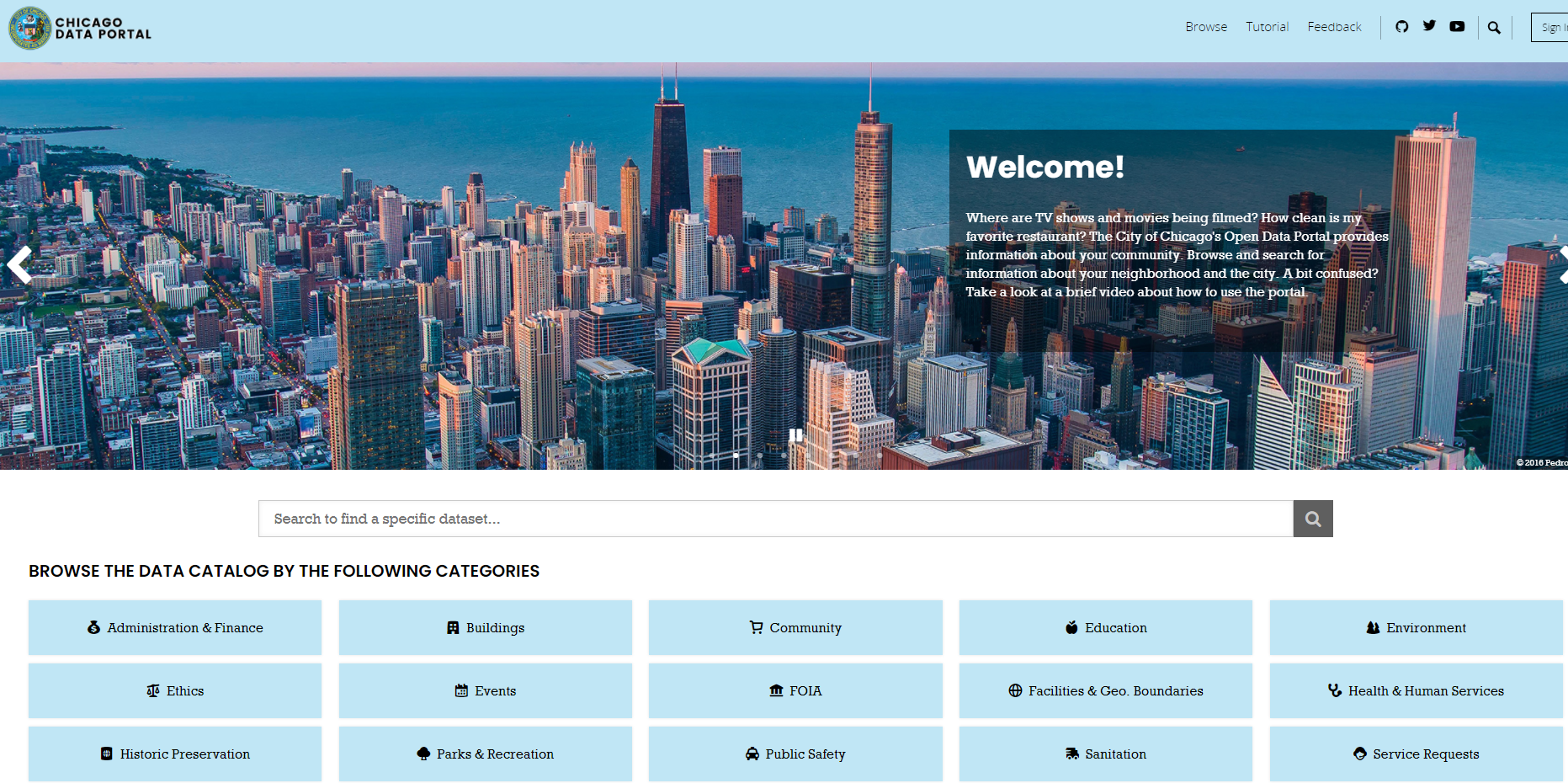
*Fin : 29/11/2021*

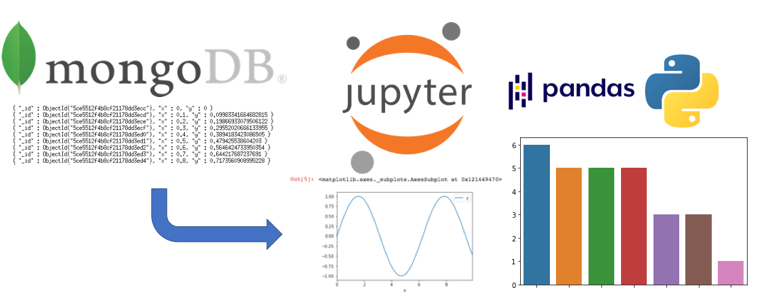
**Sprint 1 - Avancement du Prototype et Migration du projet sur le serveur du CNAM**

**Rappel du Projet et des avancements du Sprint 0**

A l’issue du Sprint T0, plusieurs étapes essentielles ont été réalisées, notamment :

* le choix de notre base de données d’étude
* la définition de notre architecture technique
* la construction de prototype d’indicateurs résumant les données de criminalité de la ville de Chicago





Les problèmes d’accès à internet depuis le lieu de l’école nous a empêché de finaliser notre Sprint T0 avec une démonstration du stockage des données dans l'écosystème Hadoop (HDFS). Cependant, en stockant les données depuis la base de données MongoDB (Base No SQL) et en les requêtant à l’aide de la librairie “PyMongo” Python, nous avons réussi à prototyper l’ensemble de nos indicateurs définis en début de Sprint.

Le Sprint 1 devait ainsi être consacré au stockage des données dans le serveur HDFS du CNAM et au requêtage Map Reduce.

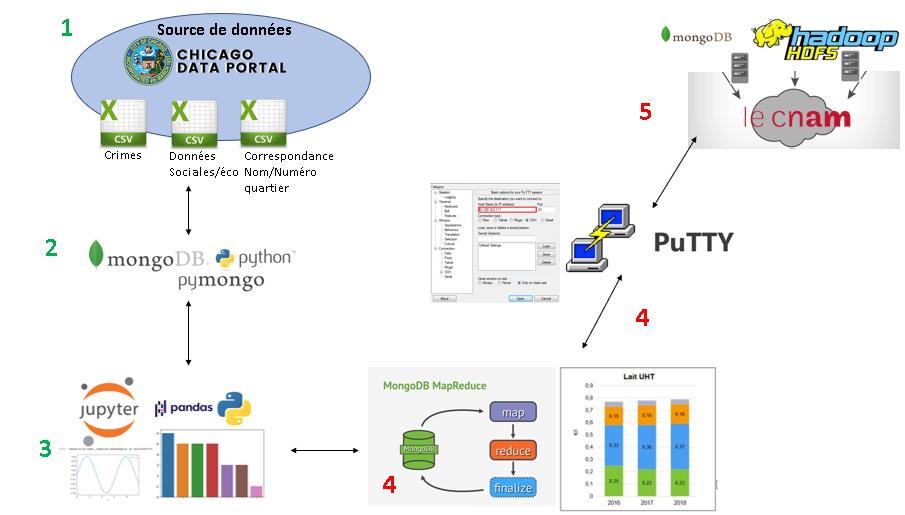
**Changement de contexte du projet**

Ce sprint 1 se caractérise par sa durée très courte entre notre retour à l’école et la présentation de ce sprint 1. A cela s'ajoutent des problématiques d’accès à Internet depuis le bâtiment de l’école (pas de ports Ethernet sur nos ordinateurs personnels) et du serveur CNAM depuis chez nous. Le non fonctionnement des outils Python sur les postes fixes de l’école posait également problème.

C’est dans ce contexte, que nous avons décidé d’être le plus efficace possible durant cette courte période et donc d’adapter notre solution technique.

En utilisant pleinement la méthode agile, nous avons ainsi décidé de poursuivre notre prototype Python développé lors du sprint 0 et de tenter d’avancer sur le développement de notre solution sur la machine Mongo DB du serveur du CNAM.

**Définition d’un nouveau processus technique pour ce sprint**



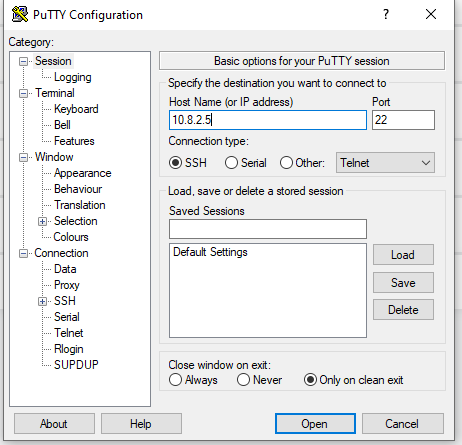
En **vert,** les étapes déjà réalisées dans le cadre du sprint 0. En **rouge** les étapes réalisées dans le cadre de ce sprint 1.

A cause des contraintes citées précédemment, nous sommes repartis sur notre prototype Python que nous complétons durant ce sprint.

**Définition du serveur et accès avec Putty**

Il existe au sein des nouveaux locaux du CNAM à Niort un serveur hébergeant plusieurs outils de travail Big Data.

Pour y accéder, nous nous sommes connectés en SSH par l’intermédiaire de la solution Putty, client SSH.



Au sein de ce serveur, plusieurs machines y sont installées dont l’éco-système Hadoop et Mongo DB.

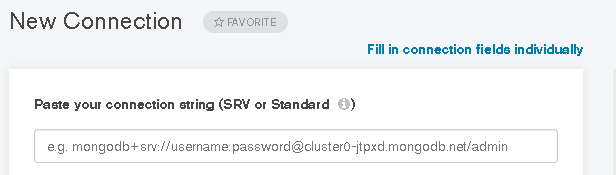




**Accès au Mongo DB du serveur et création de notre base de données de crimes**

Comme expliqué précédemment, la machine Mongo DB a été choisie par rapport au temps restant et les ressources à nos dispositions.

La connexion au serveur se fait depuis le client Mongo Compass, client choisi dans le cadre de ce projet. Dans le cadre ci-dessous de connexion,

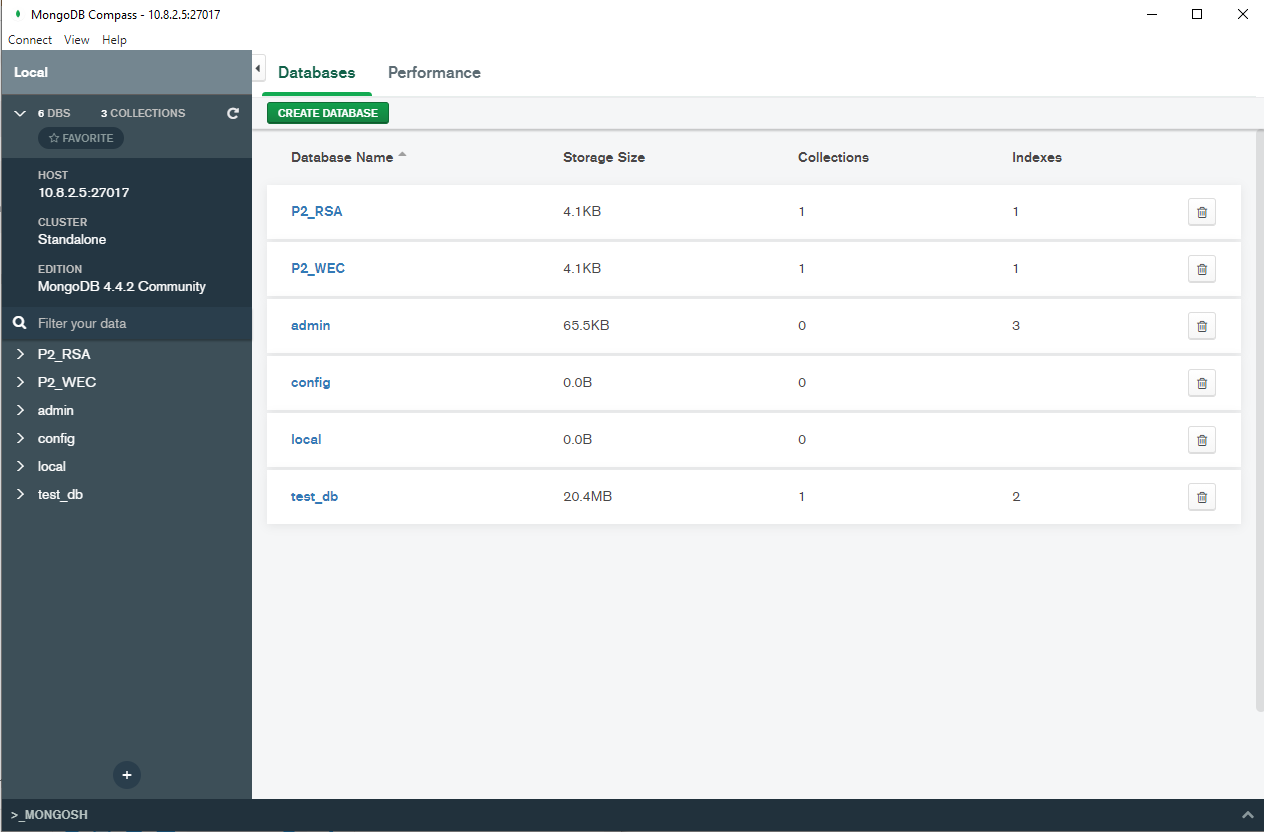


Nous y avons rentré la chaîne de connexion suivante :

mongodb://admin:passadmin33@10.8.2.5:27017/?authSource=admin&readPreference=primary&appname=MongoDB%20Compass&directConnection=true&ssl=false

Nous permettant de nous connecter sur la machine Mongo DB du CNAM.

Comme le montre la capture ci-dessous, nous y avons créé notre base de données de crimes avec nos fichiers sources csv, chacun correspondant à une collection.

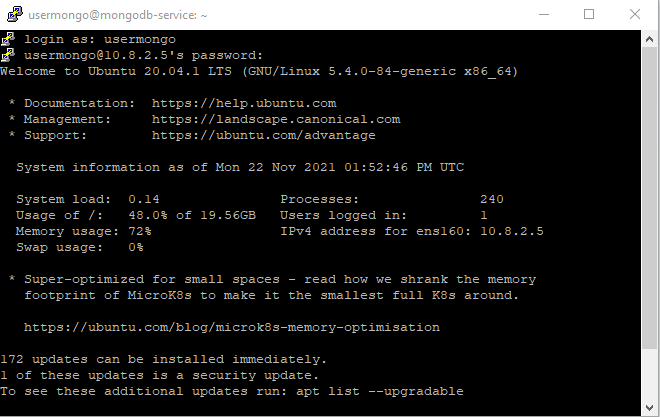


Notre base de données appelée “P2\_RSA” fait donc bien partie de la machine Mongo DB du serveur du CNAM au même titre que la base de données du groupe “P2\_WEC”

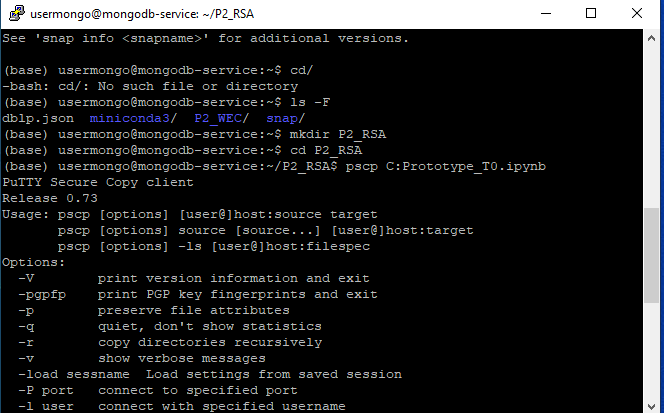
**Importation du code Python/données sur le serveur**

Une fois la connexion faite depuis la machine Mongo DB, notre objectif suivant est d’importer nos données, l’environnement Python et notre script associé dans le serveur Mongo DB du CNAM.

On se connecte sur la machine Mongo DB depuis Putty en fournissant le nom utilisateur et le mot de passe. Cela nous permet de tomber sur l’interface suivante :



Pour pouvoir nous différencier des autres projets utilisant également la machine Mongo Db du serveur, nous avons créé un dossier à notre nom dans lequel nous avons transféré nos codes sources et nos données.



Pour celà, nous avons lancé la commande suivante :

pscp "C:\**crimeT1.py**" usermongo@10.8.2.5:P2\_RSA

avec en **bleu,** le fichier à importer dans le serveur.

Nous avons utilisé **la commande PSCP**. Il s’agit d’un client SCP associé à l’outil PuTTY, est un outil pour nous permettre de transférer des fichiers de façon sûre d'une machine à une autre, via une connexion SSH.

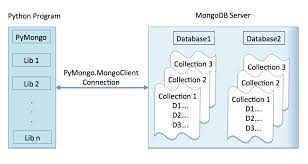
Nous nous retrouvons ainsi avec notre dossier constitué de notre script Python et de nos données utiles pour notre étude.



**Développement requêtes Map Reduce avec Python**

Pour rappel, l’un de nos deux objectifs de ce projet est de résumer la complexité des crimes de Chicago à la fois pour les aspects temporels (évolution du nombre de crimes), géographiques (nombre de crimes par area) et structurels à travers différents indicateurs.

Pour pouvoir réaliser cette étape, nous avons repris notre fichier Python du Sprint 0 dans lequel nous avons réalisé des visualisations graphiques pour mieux comprendre nos données de crimes.

A l’aide de la librairie Python “Pymongo”, il est possible de requêter et de calculer des indicateurs depuis des bases de données enregistrées en localhost sur MongoDB. Nous avons décidé de choisir cette méthode pour optimiser nos indicateurs d'agrégation sur la criminalité à Chicago 

L'API de PyMongo prend en charge toutes les fonctionnalités du moteur map/reduce de MongoDB. Une fonctionnalité intéressante est la possibilité d'obtenir des résultats plus détaillés lorsque vous le souhaitez, en passant full\_response=True à [map\_reduce()](https://api-mongodb-com.translate.goog/python/2.0/api/pymongo/collection.html?_x_tr_sl=en&_x_tr_tl=fr&_x_tr_hl=fr&_x_tr_pto=nui,sc#pymongo.collection.Collection.map_reduce) . Cela renvoie la réponse complète à la commande map/reduce, plutôt que simplement la collection de résultats.

**Map reduce** est un paradigme de traitement de données pour condenser de grands volumes de données en résultats agrégés utiles. En termes très simples , la commande MapReduce depuis l’API PyMongo prend deux entrées primaires , la fonction de mappage et la fonction de réducteur .

Un Mapper débutera par la lecture d'une collection de données pour construire une map avec les champs que nous souhaitons traiter. Puis cette paire clé, valeur est introduite dans un réducteur , qui traitera les valeurs.

Pour faire simple, la fonction **map** va émettre (*emit*) des paires clé/valeur sur lesquelles **reduce** va travailler. Le “reduce” va faire la moyenne du nombre de crimes pour un “location Description” ou un “Primary Type”. Enfin, une collection est créée pour abriter le résultat de l’exécution du map-reduce.

Le premier calcul Map/Reduce qui nous intéresse a été réalisé durant ce sprint 1 :

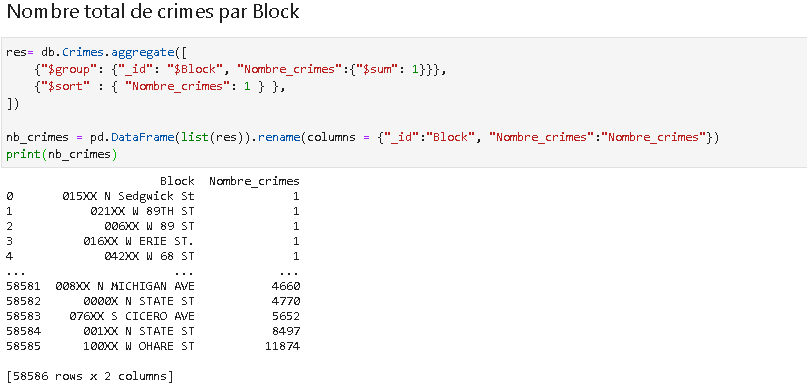
* Nombre de crimes par “Location Description” (simple), quartiers de la ville

Nous avons privilégié cette requête car en se renseignant sur Internet, nous avons appris que les lieux les plus reconnus de Chicago sont les lieux où la criminalité est la plus forte.

Notre volonté était donc d’analyser cette information avant de pouvoir analyser son pouvoir prédictif sur la criminalité, étape réalisée lors du prochain sprint.

Cette requête a tout d’abord été lancée depuis Python, réalisée à l’aide de la librairie “PyMongo”. Nous nous sommes uniquement appuyés sur un simple “aggregate”

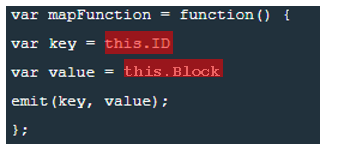
Une capture ci-dessous montre le script lancé et le tableau des résultats récupéré en retour



La requête n’a donc plus été lancée depuis notre machine mongo DB en local mais depuis la machine Mongo du serveur du CNAM.

Puis cette requête a été lancée depuis le client Compass de Mongo DB en employant cette fois-ci le paradigme Map Reduce expliqué précédemment.

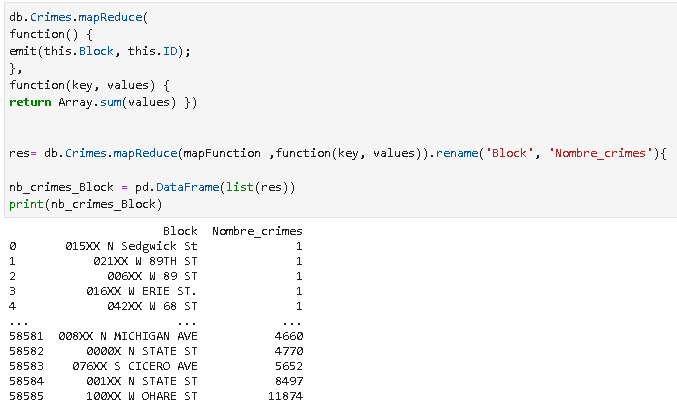
Tout d’abord on agrège notre ID identifiant les crimes. Puis un compteur d’ID est tourné pour chaque “Block”.





Enfin, l’appel à la fonction mongo “.Map reduce” nous permet de récupérer les résultats sous forme de table à deux entrées avec le nombre de crimes par Block.

Nous retrouvons exactement le même tableau obtenu avec l’outil Python et la requête d’agrégation/Map reduce.



**Conclusion Sprint 1**

A l’issue du Sprint T1, plusieurs étapes ont été réalisées, notamment :

* Evolution de la définition de notre processus technique en mode agile
* Lancement de requêtes Map Reduce en local depuis une base Mongo DB
* Accès au serveur du CNAM et import de l'écosystème de notre projet dans la machine Mongo DB

Les problèmes externes au projet et la faible durée du sprint nous a empêché de finaliser notre Sprint T1 convenablement avec une démonstration du Stockage des données dans l'écosystème Hadoop (HDFS).

Cependant, en stockant les données depuis la machine Mongo DB du serveur du CNAM et en les requêtant à l’aide de la librairie “PyMongo” Python et du client Compass, nous avons réussi à prototyper la création d’un de nos indicateurs à l’aide de deux différentes manières.

Nos objectifs fixés en début de Sprint sont donc atteints. De plus, la méthode Agile employée dans le cadre de ce projet nous a permis de rebondir face à une problématique d’accès à internet en définissant notamment un prototype de calcul d’indicateurs.

A partir de ce sprint, l’ensemble de nos démarches vont converger vers le serveur et la machine Mongo DB pour automatiser nos résultats.

**Perspectives pour le Sprint 2**

Nos prochaines étapes sont :

* Prédiction d’indicateurs grâce à divers algorithmes de Machine Learning. Prédiction de la sévérité du crime (grave ou non) et du type de crimes (vol, agression etc …)
* Réalisation d’un tableau de bord/interface graphique (visualiser en 1 seul coup d'œil nos indicateurs de visualisation et de prédiction). Il s’agira d’une interface graphique qui constituera un bilan de nos activités du projet

L’environnement de travail sera également sur Python en poursuivant nos démarches du Sprint 0 et 1.

Ci-dessous, la backlog mise à jour à l’issue de ce Sprint T1 :

