

Projet Octoplus

Certifications : développer une base données - exploiter une base de données

Simplon - Formation Développeur Data - Fabrique Élancourt

Préparé par : Gérald Bouget

15 juillet 2020

PRÉSENTATION

Les statistiques sur la criminalité sont l'objet de discussions particulièrement passionnées compte

tenu des enjeux de société qu'elles peuvent traduire : la violence est-elle en augmentation ?

Comment sont réparties les infractions commises sur le territoire français ?

Dans quel département ou dans quelle région le nombre d'infractions est le plus important ?

Ces questions sont passionnantes et il m'a donc semblé intéressant de pouvoir étudier les données disponibles s'y rapportant tout en mettant à profit mes compétences acquises pendant la formation Développeur Data.

Il existe un jeu de données open data principal qui recense tous les crimes et délits enregistrés par les services de police et de gendarmerie en France depuis 2012 et qui est disponible sur la plateforme [datagouv.fr](https://www.datagouv.fr) depuis 2015.

Buts

Ce dataset au format excel, réparti sur de nombreux onglets multi-indexés est composé de données très intéressantes puisqu'il permet notamment de savoir quel est le nombre de chaque type d'infractions par département et par année ainsi que par service de police ou de gendarmerie.

Cependant pour en tirer pleinement profit il est nécessaire de le restructurer.

Le but est de pouvoir mettre à disposition ces données pour tous publics de manière compréhensibles, intégrées et visuelles.

Les crimes et délits touchent des questions de société importantes. Il me semble donc aussi essentiel que les données de ce dataset puissent être mises en perspective avec des données externes socio-économiques et notamment démographiques.

Solution

La solution proposée pour atteindre ce but est la création d'une base de données relationnelle évolutive qui sera composée des données du dataset ci-dessus référencé mais aussi de données externes démographiques et informationnelles. Cette base de données sera directement interrogable à partir d'un dashboard interactif disposant de multiples critères de sélection et de visualisation sous forme de carte et de graphiques.

J'ai nommé ce projet 'Octoplus' en référence au huit premières tables qui comptaient la base de données que j'ai conçu et aussi parce qu'il est destiné à être évolutif.

Contour du projet

La limite de ce projet sera de réaliser toute la partie développement et exploitation de la base de données. Une partie déploiement de la base de données sur serveur distant et du dashboard en ligne est inclue dans le projet mais n'en faisait pas partie initialement dans le cahier des charges

Ce rapport va ainsi permettre d'expliquer tous le processus de réalisation à la fois conceptuel, technique et de gestion de projet.

TABLE DES MATIÈRES

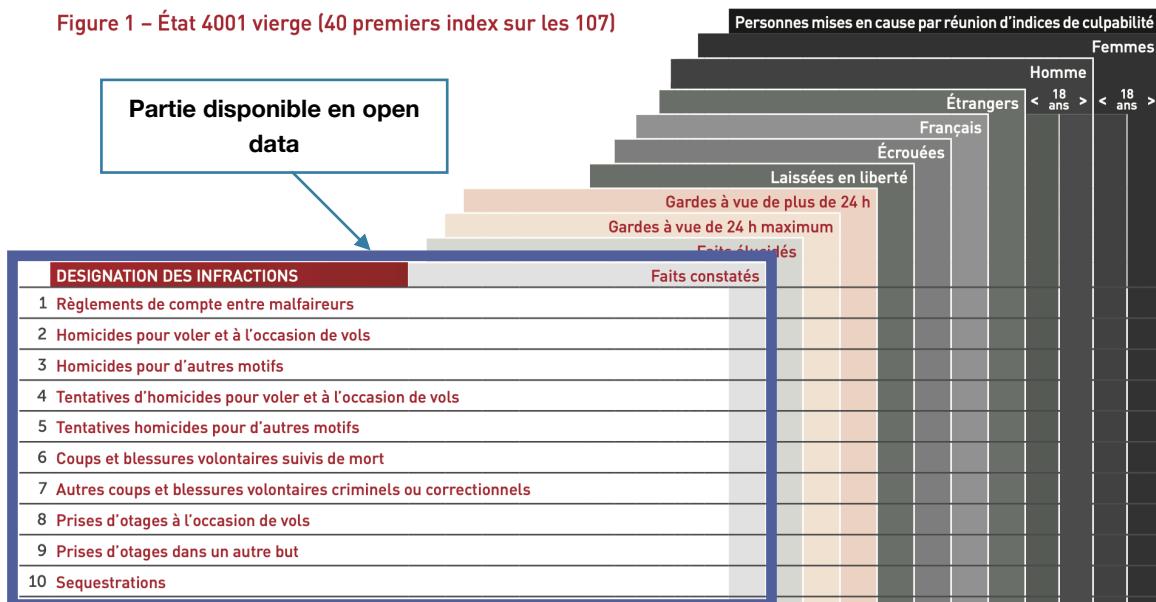
PARTIE 1 : ANALYSE CONCEPTUELLE DU PROJET ET CAHIER DES CHARGES	5
I - Contexte de la collecte des données sur la délinquance	5
II - Une source de donnée open data peu lisible en l'état	7
III - Besoin identifié : rendre ces données accessibles facilement via une base de données accompagnée d'un dashboard	9
IV - Avertissement sur l'interprétation des données	9
V - Moyens techniques envisagés	10
VI - planning prévisionnel	12
PARTIE 2 : CONCEPTION ET DÉVELOPPEMENT DE LA BASE DE DONNÉES	13
I - Conception de la base de données (MCD et MPD)	13
II - Data cleaning et préparation des tables	15
A. PROCESSUS DE CONTRÔLE PRÉALABLE DES DOUBLONS ET COHÉRENCE DES DONNÉES DANS LE DATASET	21
B. STRUCTURATION DES TABLES DU LA BASE DE DONNÉES AVEC PYTHON/PANDAS	23
III - Création de la base données et optimisation	28
A. DÉFINITION DES CLEFS PRIMAIRES	28
B. EXEMPLES DE CRÉATION DES TABLES PRINCIPALES	29
IV - Importation des données dans la base	29
V - Ajouts contraintes et optimisation de la base données (backup, automatisation,...)	30
A. AJOUT DES CONTRAINTES	30
B. VÉRIFICATION DE LA CRÉATION DES TABLES, INSERTION DES DONNÉES ET AJOUT DES CONTRAINTES	30
C. SAUVEGARDE AUTOMATISÉ DE LA BASE DE DONNÉES AVEC CRONTAB	31
D. APPLICATION PYTHON POUR MISE À JOUR AUTOMATISÉ DE LA BASE DE DONNÉES ET DE CONTRÔLE DES DONNÉES AVANT IMPORT	32
E. AUTRES OPTIMISATION DE LA BASE DE DONNÉES	32
F. AJOUT TRIGGERS POUR GESTION D'ERREURS D'INSERTION ET D'UPDATE	34
PARTIE 3 : EXPLOITATION DE LA BASE DE DONNÉES	35
I - Le concept	35
II - Choix technique et méthode adoptée - conception	37
III - Rendu visuel	37
III - Déploiement du dashboard en ligne	39
A. DÉPLOIEMENT DE LA BASE DE DONNÉES OCTOPLUS SUR RDS (AWS)	39
B. DÉPLOIEMENT DE L'APPLICATION EN LIGNE SUR HEROKU	40
PARTIE 4 : BILAN ET AMÉLIORATIONS POSSIBLES	40
CONCLUSION	41
REMERCIEMENTS	41
BIBLIOGRAPHIE	42
ANNEXES	43

PARTIE 1 : ANALYSE CONCEPTUELLE DU PROJET ET CAHIER DES CHARGES

I - Contexte de la collecte des données sur la délinquance

La statistique de la police et de la gendarmerie prend son essor dans les années soixante-dix lorsque le service central d'étude de la délinquance (SCED) de la direction centrale de la police judiciaire (DCPJ) met en place en 1972 un outil de suivi statistique des crimes et délits constatés par les forces de sécurité : l' « état 4001 », du nom du formulaire administratif associé qui est transmis par chaque service élémentaire (environ 7 000) à la DCPJ. Il s'agit d'un tableau ([annexe](#)) composé de 107 lignes et 12 colonnes. Les infractions y sont classées en 107 catégories nommées index (dont 103 sont utilisés), très hétérogènes par la nature et la gravité des faits.

Figure 1 – État 4001 vierge (40 premiers index sur les 107)



La mesure statistique de La déLinquance - par Alexandre Estival - AJ Penal - Avril 2019

Les données statistiques récoltées par la DCPJ recensent uniquement l'**enregistrement des seuls délits et crimes portés à la connaissance des services de police de compagnies de gendarmerie**.

La comptabilisation statistique s'effectue selon des **unités de compte** (victime, victime entendue, plaignant, infraction...) spécifiques à chacun des index de la procédure. Ces unités de compte fixent les modalités de comptage au moment de l'enregistrement.

Lorsqu'une procédure comporte plusieurs infractions identiques, la statistique de police compte '1' et rapporte à l'index correspondant de la nomenclature de l'État 4001. Si la procédure comporte plusieurs infractions différentes, selon la gravité et chaque infraction et de leur distinction, on va retenir une seule infraction ou plusieurs.

Cela signifie que le nombre total de faits constatés est une somme de procédures, d'infractions et même de véhicules et même d'auteurs (en cas d'infraction sur les stupéfiants).

La collecte des données est réalisé à chaque fois qu'est établi une **procédure judiciaire** dans laquelle est consignée une infraction de crime ou de délit.

La collecte est centralisée au niveau de chacune des directions de police nationale et de la gendarmerie nationale. Il revient ensuite à ces services de transmettre par voie informatique leurs fichiers statistiques à la DCPJ qui est seule en charge de leur compilation.

À partir de cette compilation une version est mise en ligne sur le site de data.gouv en open data. Ce dataset est intitulé '**Crimes et délits enregistrés par les services de police et de gendarmerie depuis 2012**' et est mise à jour une fois par an en janvier

Données Réutilisations Organisations Tableau de bord Documentation

Connexion / Inscription

Recherche

Crimes et délits enregistrés par les services de gendarmerie et de police depuis 2012

Ce jeu de données provient d'un service public certifié 

Ces données constituent une actualisation des volumes Criminalité et délinquance constatées en France édités par la Direction centrale de la police judiciaire, publiés jusqu'en 2013 à la Documentation française, et disponibles sur son site [Internet](#). Elles contiennent des informations, de caractère administratif, sur l'activité judiciaire des services de gendarmerie et de police, y compris celles des DOM-COM, depuis 2012.

Ressources

[crimes-et-delits-enregistres-par-les-services-de-gendarmerie-et-de-police-depuis-2012.xlsx](#) (3.4Mo) 543 Disponible

[PRÉVVISUALISER](#) [TÉLÉCHARGER](#) 

[Documentation des chiffres annuels relatifs aux crimes et délits enregistrés par les services de gendarmerie...](#) (1.1Mo) 92 Disponible

[TÉLÉCHARGER](#) 

Producteur


Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'INTÉRIEUR

Ministère de l'Intérieur

Placé au cœur de l'État, le ministère de l'Intérieur assure la permanence et la continuité de l'État. Cette fonction régaliennes se concrétise par le rôle majeur et les services...

[VOIR LE PROFIL](#) 

II - Une source de donnée open data peu lisible en l'état

Le dataset disponible est un fichier excel d'environ 3.6 Mo dont l'ouverture via un tableur est très longue.

Extrait du dataset

Il est composé de 18 onglets, avec une structure différente selon qu'il s'agit des données de police ou de gendarmerie. Chaque feuille est multi-indexée au niveau de l'index et des variables. Le nombre de colonnes est différent pour chaque onglet et d'une année à l'autre le nom et le

nomOnglets	nombreColonnes	nombreLignes	nombreCellules
Services PN 2012	609	110	66990
Services PN 2013	604	110	66440
Services PN 2014	599	110	65890
Services PN 2015	733	110	80630
Services PN 2016	683	110	75130
Services PN 2017	680	110	74800
Services PN 2018	680	110	74800
Services PN 2019	681	110	74910
Services GN 2012	383	109	41747
Services GN 2013	385	109	41965
Services GN 2014	382	109	41638
Services GN 2015	382	109	41638
Services GN 2016	379	109	41311
Services GN 2017	379	109	41311
Services GN 2018	379	109	41311
Services GN 2019	379	113	42827
Total	8317	1756	913338

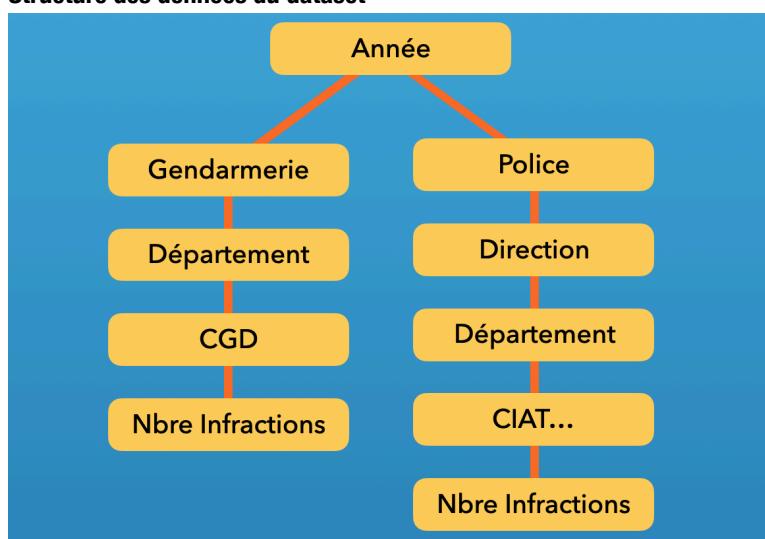
Statistiques dataset (réalisé via script sur un notebook Jupyter)

nombre des services évolue.

La granularité la plus fine correspond à la variable 'département'.

Le dataset est donc peu exploitable en l'état et a besoin d'être restructuré pour pouvoir en tirer tout son potentiel.

Structure des données du dataset



III - Besoin identifié : rendre ces données accessibles facilement via une base de données accompagnée d'un dashboard

Le but est de pouvoir rendre ces données lisibles pour toute personne intéressée par ces informations que ce soit un citoyen quelconque, un journaliste ou une institution publique.

La solution proposée est la suivante:

- la création d'une base de données relationnelle qui permettra de préserver une intégrité des données et un requêtage rapide.
 - la création d'un Dashboard interactif, relié à la base de données et interactif qui permettra la visualisation de données à partir de critères sélectionnés par l'utilisateur.

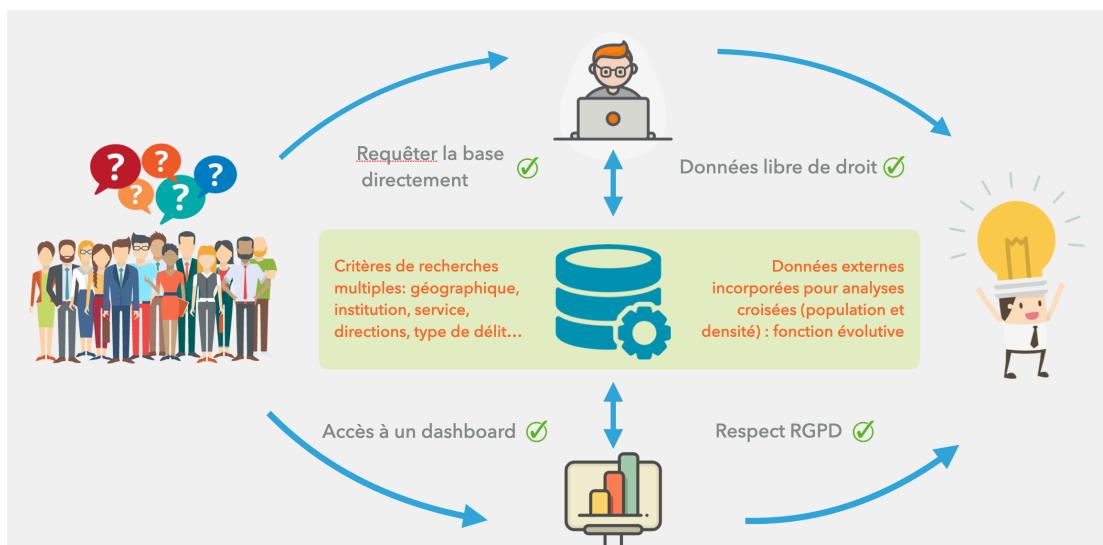


Schéma fonctionnel du projet

La base de données sera complétée de données externes suivantes pour permettre une meilleure compréhension des données ainsi qu'une mise en perspective avec des données socio-économiques :

- **Les indicateurs** (catégorie d'infractions créée par la SSMSI¹)
 - **Les unités de compte** (ce qui est pris en compte pour comptabiliser une infractions)
 - **Les noms de département** (seuls les numéros sont renseignés dans le dataset)
 - **Des données démographiques** (densité de population, nombre de la population...)

¹ Service statistique ministériel de la sécurité intérieure (SSMSI)

Composé notamment de statisticiens de l'INSEE, de policiers et de gendarmes, le SSMSI a expertisé les données de l'état 4001 pour sélectionner une série de **12 indicateurs qui s'appuient sur les index de l'état 4001**.

Les indicateurs présentés ont été choisis et construits à partir des informations disponibles dans les bases de données des procédures, en fonction de plusieurs critères :

- La pertinence du phénomène observé
- La cohérence entre les données administratives et les résultats des enquêtes de victimisation
- La stabilité dans le temps des chiffres, preuve de la fiabilité de leur mode de production et de construction

Ces indicateurs sont importants car ils permettent de se référer à des données qui sont relativement fiables.

IV - Avertissement sur l'interprétation des données

Les données issues de ce dataset sont à utiliser et interpréter avec prudence.

En effet les éléments suivants sont à prendre en compte et sont précisés dans la documentation jointe au dataset:

- Les systèmes d'enregistrement des infractions ont beaucoup évolué ces dernières années : à la gendarmerie nationale en 2012 et en 2015, à la police nationale depuis 2013. Une partie des évolutions dans les chiffres correspond donc moins à l'évolution réelle des phénomènes qu'à celle des pratiques d'enregistrement.
- la liste des services de gendarmerie et de police évolue au fil des ans et certains services peuvent être créés, supprimés, ou voir leur territoire de compétence être modifié d'une année à l'autre. C'est pourquoi les évolutions des chiffres d'un même service peuvent ne pas être facilement interprétables. Les modifications de territoires de compétence des services sont publiées au Journal Officiel.
- Ruptures statistiques constatées par le SSMSI (<http://www.interieur.gouv.fr/Interstats>) : les modalités d'enregistrement des crimes et des délits évoluent dans le temps. De ce fait, les comparaisons entre périodes ne sont pas toujours pertinentes, ou doivent être réalisées avec précautions. Il est important, avant de tirer des conclusions des chiffres ci joints, de tenir compte des ruptures techniques constatées ces dernières années ou ces derniers mois.
- Les analyses menées par la SSMSI sur ces données issues de la police et de la gendarmerie, sont accompagnées de données issues d'une enquête menée directement auprès de la population pour mesurer le degré de sentiment d'insécurité. Cette enquête permet de réaliser une analyse plus fine de la délinquance mais les résultats

ne sont pas disponibles de manière exhaustives et ne seront donc pas intégrées à ce projet.

CHOIX DES OUTILS

SQL Alchemy	- Connexion à la base données MySQL dans un environnement Python	- ORM performant
Plotly / ipyWidget	- Visualisation des données	- Larges options - Interactivité - Implémenté dans Dash -ipydiget : pour interaction dans Jupiter notebook / Prévisualisation pour Dash
Dash	- Dashboard	- Utilise python - Gratuit - Performant pour l'interactivité
Open Office	- Ouverture et lecture du dataset	- Gratuit - Mêmes fonctionnalités que Excel en lecture
Jupyter Notebook	- Préparation des données -Exploitation des données - Visualisation	- Distribution performante et souple pour utiliser avec Python
Atom	- Scripts SQL Applications /Dashboard	- Gratuit et habitué - Interface Git et terminal
UML	- MCD	- Interface ergonomique
Git	-utiliser pour versionning	- suivi version et performant
Conda Env.	-Pour applications	-Permet pérennité du projet et cross platform
Outils	Utilisation	Raison
MySQL (V8.0.18)	- Conception de la base de données	- Open source et gratuit - Documentation importante - Adapté pour une utilisation web
MySQL Workbench	- Modélisation MPD - Utilisation Procédures stockées	- Interface intuitive - Gratuit - Inclu dans le package MySQL
Python (V 3.7)	- Tout au long du processus de préparation / structuration des données / Exploitation / visualisation	- Bonne maîtrise du langage - Librairies adaptées à mon projet
Openpyxl / (librairie Python)	- préparation et structuration des tables	- Format du dataset en excel - Large possibilités de manipulation de données en excel - utiliser conjointement avec pandas.ExcelFile / pandas
Pandas (librairie python)	- préparation / nettoyage des données - exploitation des données - Requêtes SQL	- Outil de base de manipulation des données - Très performant - Facile d'utilisation

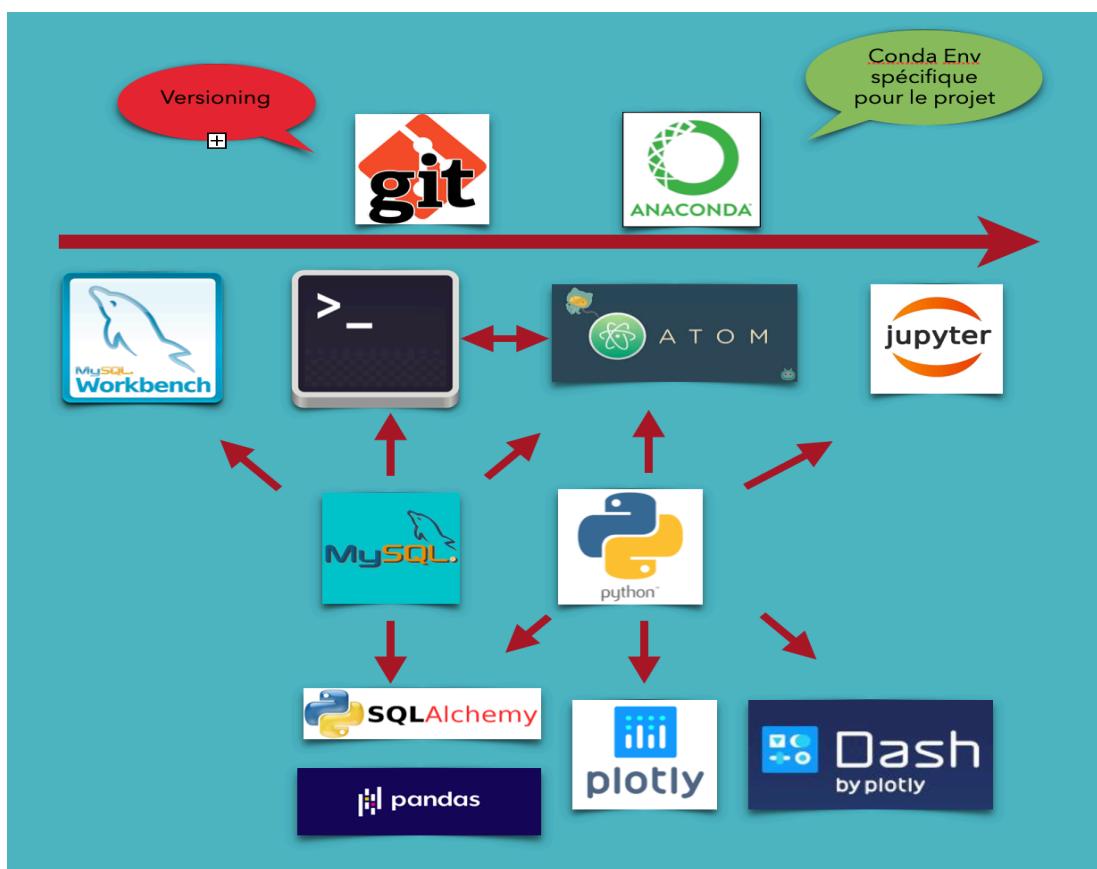
V - Moyens techniques envisagés

Le projet final comportera deux applications principales et une secondaire:

- Une base de données relationnelle
- Un dashboard connecté à la base de données
- Une application python pour le contrôle des mises à jour en ligne du dataset, et pour la mise à jour automatique de la base de données.

Pour gérer les multiples versions de code, j'ai utilisé le logiciel de gestion de version décentralisé Git. Concernant l'application Dash, j'ai également utilisé un environnement virtuel spécifique pour éviter des conflits de versions éventuels avec mon environnement global et permettre une utilisation cross platform.

Les principaux outils utilisés pour le projet



Pour accéder facilement et rapidement aux différents répertoires et applications du projet, j'ai créé quelques alias en Bash sur mon terminal:

```
export EDITOR=nano
alias dash_rep='cd /Users/geraldbouget/SFR\ Cloud/Documents/Simplon_DataDev_formation/Projet_final/dashboard'
alias dashboard='atom; cd /Users/geraldbouget/SFR\ Cloud/Documents/Simplon_DataDev_formation/Projet_final/dashboard; conda activate dash_env'
alias mybase='cd /usr/local/mysql/bin; ./mysql -u root -p'
alias jupyter_pj='cd /Users/geraldbouget/SFR\ Cloud/Documents/Simplon_DataDev_formation/Projet_final/Notebooks_A_UTILISER_; jupyter notebook'
alias majauto='cd /Users/geraldbouget/SFR\ Cloud/Documents/Simplon_DataDev_formation/Projet_final/automatisation; conda activate octo_maj_env'
alias octonb='cd /Users/geraldbouget/SFR\ Cloud/Documents/Simplon_DataDev_formation/Projet_final/Notebooks_A_UTILISER_/'
(base) MBP-de-Gerald:~ geraldbouget$
```

VI - planning prévisionnel

La part essentielle du travail a été réalisé sur une période d'un mois environ. Elle comprenait:

- L'analyse des besoins et le choix du dataset
- le développement de la base de données (MCD, data cleaning, codage base, import des données)
- l'exploitation de la base sur notebook

Les délais ont été très serrés et un peu dépassés dans certains cas principalement sur la partie data cleaning/ préparation des tables.

Pour la partie 'préparation des données' il a fallu faire face à des erreurs dans le dataset et donc prendre du temps pour s'assurer de la cohérence et de l'intégrité des données.

Pour la partie exploitation il a fallu prendre un peu plus de temps que prévu pour maîtriser au mieux la librairie 'Plotly' .

MACRO - PLANNING INITIAL VS RÉALISÉ

7/05 - 10/05	10/05 - 20/05	20/05 - 31/05	01/06 - 05/06	10/06 - 30/06
Analyse des besoins et des objectifs	Développement de la base de données	Exploitation de la base de données	Préparation 'prez' client + marge de sécurité	Optimisation + dashboard
Analyse des besoins et des objectifs	1. Modélisation 2. Préparation données 3. Création tables et import 4. Contraintes, optimisation 5. Sauvegarde	1. Requêtes et vérification cohérence des données 2. Analyse de données 3. Visualisation via plotly dans NB	1ere 'prez' client + finitions	1. App automatisation backup et MAJ 2. optimisation BD 3. Dashboard Dash

J'ai utilisé Trello comme outil de gestion. Les semaines étaient organisées sur des mini-sprint de 4 jours et un dernier jour de code review et de marge pour finir le sprint.

PARTIE 2 : CONCEPTION ET DÉVELOPPEMENT DE LA BASE DE DONNÉES

I - Conception de la base de données (MCD et MPD)

La granularité la plus fine du dataset, avec le département, est le ‘service’ (un commissariat ou une compagnie de gendarmerie par exemple). Chacun de ces service est relié à un département unique et ils comptabilisent pour chaque infraction du Code Index le nombre d’infractions enregistrée sur une année. Chaque département peut contenir plusieurs services mais un service est lié à un seul département.

Chaque année le nombre et le nom des services peut varier. Certains sont supprimés et d’autres modifiés.

Pour la police, il est également précisé de quelle direction relève chaque service. Chaque service relève d’une seule direction mais une direction peut diriger plusieurs services. Le nombre de ces directions n’est pas fixe selon les années non plus.

Les département sont identifiés par leur numéro et leur nombre est exhaustif et identique chaque années.

De même le nombre et la liste des infractions est une nomenclature figée, identique chaque année et pour chacune des institutions police ou gendarmerie.

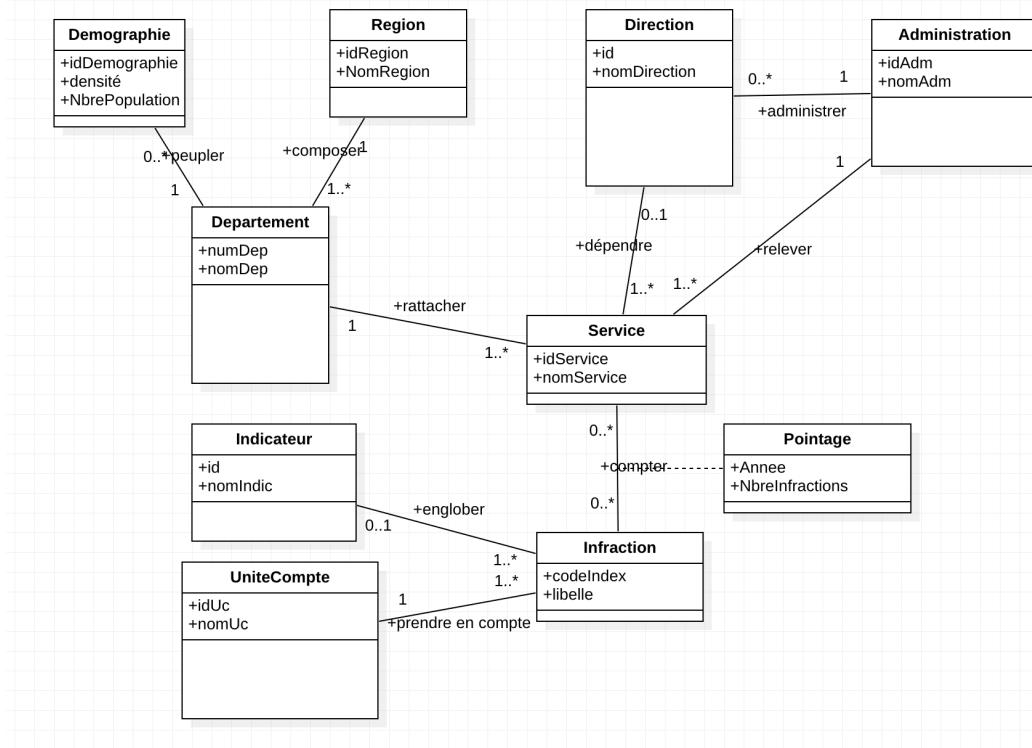
Pour chaque année, il y deux onglet excel qui sont créés: un pour les crimes et délits enregistrés par la police et un pour ceux enregistrés par la gendarmerie.

Les 4 classes principales sont :

- la classe Service
- la classe Infraction
- la classe Pointage
- et la classe Direction

La classe **Pointage** est une classe-association issue de la relation ‘plusieurs à plusieurs’ entre Service et Infraction. Chaque service comptabilise 0 ou plusieurs infractions et chaque infraction est comptabilisé par 0 ou plusieurs services. La classe Pointage est une classe intermédiaire qui va correspondre au nombre des infractions comptabilisées. Elle va devenir une table part entière identifié par les deux clés primaires issues de la table Infraction (codeIndex) et de la future table Service (idService).

Modèle conceptuel de données (MCD notation)



J'ai ajouté l'attribut 'année' afin de pouvoir identifier les années d'enregistrement des infractions, sachant que ce pointage est annuel. J'ai donc créée cette classe à part. De chaque direction va dépendre un service et chaque service dépend de 0 ou une direction.

La classe **Administration** permet de distinguer les services de gendarmerie ou de police. Elle est à la fois reliée à la classe direction et service car les services de gendarmerie n'ont pas de direction pour l'instant mais cela pourrait arriver et j'ai donc anticiper.

La liaison directe entre Service et Administration permettra aussi de simplifier certaines requêtes quand il faudra identifier les services relevant de la gendarmerie.

La table 'Direction' correspond à la liste unique des directions dont dépendent les services de police. Les services de gendarmerie n'en disposent dans ce dataset mais pourront éventuellement être précisés à l'avenir.

Chaque service est rattaché à un département unique. Dans le dataset ces départements sont identifiés par leurs identifiants officiels (75, 86...).

Dans un même département on trouve plusieurs services de gendarmerie et de police mais concernant les Dom-Tom la liste des départements diffèrent légèrement entre police et gendarmerie. De plus il me semblait intéressant pour la visualisation des données d'avoir le nom correspondant des départements. Les liste des département et leur nomenclature est sujet à d'éventuels

modifications dans l'avenir. Pour toutes ces raison j'ai donc créé une classe '**Departement**' relié à la classe service.

La classe '**région**' va correspondre à des données importées. Il m'a également semblé intéressant en vue de l'exploitation des données de pouvoir choisir entre une visualisation départementale ou régionale des analyses. Chaque département appartient à une région et chaque région est composée de un ou plusieurs départements .

La classe '**Demographie**' correspond également à des données externes qui enrichiront les analyses ultérieures. Les données correspondantes sont déjà agrégées par département et proviennent de l'INSEE.

Mais les données ne dépendent pas uniquement du département mais aussi de l'année de recensement. J'ai donc créé une classe à part qui aura au final une clé primaire composite (numéro de département et année de recensement).

Les classes '**indicateur**' et '**unité de compte**' sont également des classes comprenant des données externes au dataset et apportent des informations complémentaires à la nomenclature des infractions. La classe 'Indicateur' notamment correspond à des catégories d'infraction identifiée par la SSMSI et serviront pour l'exploitation des données. Il sera plus simple de se servir de ces catégories que de choisir à chaque fois entre 103 infractions ou de créer des catégories artificielles .

Voici les dictionnaires de données final après ajout de données démographiques supplémentaires sur la page suivante.

DICTIONNAIRE DES DONNÉES INITIAL

Attribut	Désignation	Type
numDep	numéro de département	CHAINE DE CARACTÈRES
densité	densité de la population au km2 pour un département	FLOAT
population	nombre de population pour un département	FLOAT
anneeRecensement	année de recensement ou de collecte des données démographiques/économiques	YEAR
IdRegion	numéro de la région (code INSEE region)	CHAINE DE CARACTÈRES
nomRegion	nom de la région	CHAINE DE CARACTÈRES
nomDep	nom du département	CHAINE DE CARACTÈRES
idDirection	numéro auto-incrémenté pour identifier chaque Direction de police	INT
nomDirection	sigle du nom de la direction de police	CHAINE DE CARACTÈRES
idAdm	numéro auto-incrémenté pour identifier chaque type d'administration (police ou gendarmerie)	INT
nomComplet	nom complet du nom des directions de police	CHAINE DE CARACTÈRES
nomAdm	nom de l'administration	CHAINE DE CARACTÈRES
idService	numéro auto-incrémenté pour identifier chaque service de gendarmerie ou police	INT
nomService	nom de chaque service de police ou gendarmerie	CHAINE DE CARACTÈRES
annee	année de collecte du nombre de dépôt de plaintes (infractions)	YEAR
codeIndex	index de référence d'une infraction donnée (index ref État 4001)	INT
libelle	nom de l'infraction (ref : État 4001)	CHAINE DE CARACTÈRES
idIndic	numéro identifiant chaque indicateur	INT
idUc	numéro identifiant chaque unité de compte	INT
nomIndic	nom de l'indicateur (catégorie rassemblant plusieurs infractions)	CHAINE DE CARACTÈRES
nomUc	nom de l'unité de compte (ce qui est pris en compte pour comptabilisé une infraction)	CHAINE DE CARACTÈRES
nomIndicLight	version moins longue du nom d l'indicateur	CHAINE DE CARACTÈRES

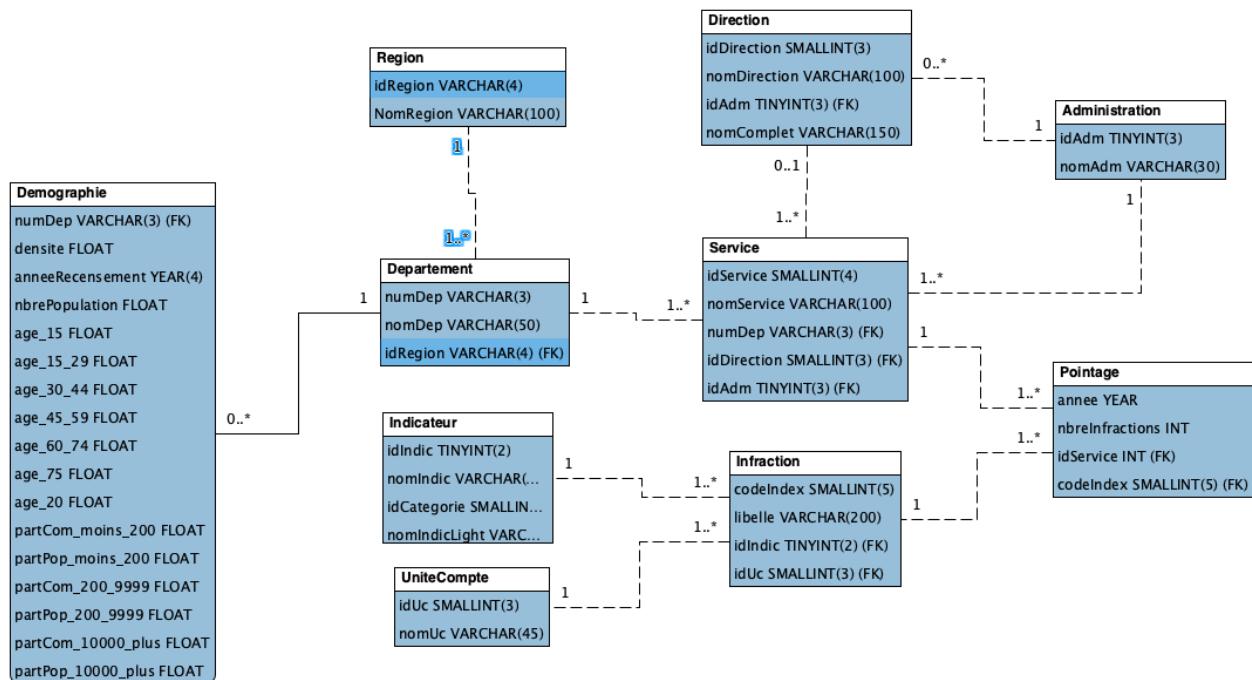
AJOUT ATTRIBUTS DÉMOGRAPHIQUES

Attribut	Désignation	Type
age_15	par de la population de moins de 15 ans en pourcentage	FLOAT
age_15_29	par de la population de 15 à 29 ans en pourcentage	FLOAT
age_30_44	par de la population de 30 à 44 ans en pourcentage	FLOAT
age_60_74	par de la population de 60 à 74 ans en pourcentage	FLOAT
age_75	par de la population de plus de 75 ans en pourcentage	FLOAT
age_20	par de la population de moins de 20 ans en pourcentage	FLOAT
partCom_moins_200	part des communes de moins de 200 habitants en pourcentage	FLOAT
partPop_moins_200	part de la population vivant dans une commune de moins de 200 habitants en pourcentage	FLOAT
partCom_200_9999	part des communes entre 200 et 9999 habitants en pourcentage	FLOAT
partPop_200_9999	part de la population vivant dans une commune de plus de 200 habitants et moins de 10 000 en pourcentage	FLOAT
partCom_10000_plus	part des communes de plus de 10000 habitants en pourcentage	FLOAT
partPop_10000_plus	part de la population vivant dans une commune de plus de 10 000 habitants en pourcentage	FLOAT

SOURCES DONNÉES EXTERNES

Données	Source
départements	https://www.data.gouv.fr/fr/datasets/r/0e016380-b70b-4443-a777-16f0bec7373e
régions	https://www.data.gouv.fr/fr/datasets/r/0e016380-b70b-4443-a777-16f0bec7373e
indicateurs	observatoire national de la délinquance et de la réponse pénale (ONDREP) et laService statistique ministériel de la sécurité intérieure (SSMSI) - Word https://www.interieur.gouv.fr/Interstats/Methode
unités de comptes	observatoire national de la délinquance et de la réponse pénale (ONDREP) et laService statistique ministériel de la sécurité intérieure (SSMSI) - Word https://www.interieur.gouv.fr/Interstats/Methode
démographie	https://www.insee.fr/fr/statistiques/fichier/2012729/TCRD_043.xls + observatoire du territoire

Modèle physique de données (MPD - notation UML)



II - Data cleaning et préparation des tables

Avant de commencer à structurer le tables, il va falloir procéder à un contrôle des données et vérifier que les points suivants sont bien respectés pour chaque onglet du dataset :

- Pour un même onglet :
 - * absence de doublons au niveau des ensembles de variables 'Département-Service' ou 'Direction-Département-Service' (variables multi-indexées)
- Pour l'ensemble des onglets :
 - * une colonne 'Code Index' numéroté de 1 à 107
 - * une colonne 'libelle' de 107 lignes identique

a. Processus de contrôle préalable des doublons et cohérence des données dans le dataset

À l'aide de la librairie Openpyxl je parse les variables Service de chaque onglet.

Je créé une fonction qui me retourne la liste des doublons et numéros de colonnes du fichier excel concerné.

```
Entrée [12]: 1 f serviceDoublons():
2     df_doublon=pd.DataFrame({
3         'nomDoubon':[],
4         'numeroColonne':[],
5         'annee':[]
6     })
7     # obligé de créer liste vide pour récuprer doublons car je parse direct colonnes je ne les verrai
8     # car df.columns va modifier nom variable
9     listeservice=[]
10    numerocol=[]
11    #nomdoublon={}
12    for name in police:
13        for services in wb[name]['C3':str(openpyxl.utils.cell.get_column_letter(wb[name].max_column)+'3')]:
14            for service in services:
15                listeservice.append(service.value)
16                numerocol.append([service.value, service.column, name])
17                nomdoublon=[item for item in services.items() if count > 1]
18                listedoublong=[nc for nc in numerocol if nc[0] in nomdoublon]
19                #countdoublon=[count[1] for count in nomdoublon]
20                if len(listedoublong)!=0:
21                    df_doublon=pd.concat([df_doublon,pd.DataFrame(listedoublong).rename(columns={0:'nomDoubon',
22                                         1:'numeroColonne',
23                                         2:'annee'}))]
24
25    listeservice=[]
26    numerocol=[]
27
28    if len(df_doublon) ==0:
29        return 'Il n\'y a pas de doublons dans les noms de service sur une année'
30    else:
31        return 'voici la liste des doublons', df_doublon
```

```
Entrée [13]: 1 serviceDoublons()
Out[13]: "Il n'y a pas de doublons dans les noms de service sur une année"
```

Pour les données 'police' pas de problème de doublons.

Pour les données 'Gendarmerie' :....

```
Entrée [13]: 1 serviceDoublons()
Out[13]: voici la liste des doublons
```

	nomDoubon	numeroColonne	annee
0	CGD LA FOA	374.0	Services GN 2017
1	CGD NOUMEA	375.0	Services GN 2017
2	CGD POINDIMIE	376.0	Services GN 2017
3	CGD LA FOA	377.0	Services GN 2017
4	CGD NOUMEA	378.0	Services GN 2017
5	CGD POINDIMIE	379.0	Services GN 2017

...on s'aperçoit qu'il y a 3 colonnes qui se répètent pour l'année 2017.

Il va donc falloir gérer ce problème au moment de la structuration des tables.

Je vérifie également qu'on a bien une liste identique d'index dans chaque onglet:

- Je vérifie d'abord que j'ai bien 107 lignes (107 codes index , nomenclature fixe).
- je crée également une fonction pour vérifier des problèmes éventuels de non concordance:

l'idée est de créer un data frame vide dans lequel je vais concaténer les deux colonnes 'Code Index' et 'libelle' de chaque année. En supprimant à chaque fois tous les doublons, seuls les signes qui ne concordent pas avec l'année précédente resteront et indiqueront qu'il y a une erreur dans le dataset. Je précise le nom de l'onglet pour retrouver facilement l'erreur.

Pour les données de police aucun problème de ce côté là.

Par contre pour les données gendarmerie on s'aperçoit qu'il y a des erreurs au sein même du dataset:

Il y en effet un décalage dans la numérotation des index. En regardant plus précisément les lignes

```
75]: 1 # check libelle et codeindex identiques pour tous les onglets
2 def checkLibelle(fichier_excel):
3     liste_probleme=[]
4     df_check=pd.DataFrame({'Code index':[], 
5                           'libelle':[]})
6     # on concat df_check avec le 1er onglet année 2012
7     # pour éviter multiindexage on ne retient que les variables service (header=1 pour gendarmerie)
8     df=pd.read_excel(fichier_excel,sheet_name=liste[0], header=1).rename(columns={'Libellé index \\ CGD':'libelle'})
9     df=df.iloc[:,[0,1]] # df avec uniquement colonne code index et libelle
10    df_check=pd.concat([df_check, df])
11
12    # on concat df_check avec les années suivantes
13    # on sépare les 2 process de concat pour que la comparaison se fasse à partir de la 2eme année
14    for sheetname in liste[1::]:
15        # on garde uniquement variable service pour éviter multiindexage (header=1 pour gendarmerie)
16        df=pd.read_excel(fichier_excel,sheet_name=sheetname, header=1).rename(columns={'Libellé index \\ CGD':'libelle'})
17        df=df.iloc[:,[0,1]] # df avec uniquement colonne code index et libelle
18        df_check=pd.concat([df_check, df])
19        if len(df_check[['Code index', 'libelle']].drop_duplicates(keep=False))>0:
20            liste_probleme.append([str(sheetname), df_check[['Code index', 'libelle']].drop_duplicates(keep=False)])
21
22    return liste_probleme
```

```
76]: 1 checkLibelle(wbpd)
: [['Services GN 2019',
  Code index                                libelle
  95      98.0                                Index non utilisé
  96     101.0                                Index non utilisé
  97  102.0  Banqueroutes, abus de biens sociaux et autres ...
  98      103.0                                Index non utilisé
  99      104.0                                Index non utilisé
 100     105.0  Prix illicites, publicité fausse et infractio...
 101      106.0          Achats et ventes sans factures
 102     107.0  Infractions à l'exercice d'une profession règl...
 103      NaN  Infractions au droit de l'urbanisme et de la c...
 104      NaN                                Fraudes fiscales
 105      NaN  Autres délits économiques et financiers
 106      NaN                                Autres délits]]
```

concernées dans le dataset, je me suis aperçu que les lignes entière avaient été décalées, et que certaines qui devaient être vides sont remplie des données de la ligne suivante.

Exceptionnellement j'ai décidé de modifier le fichier excel original directement et de prévoir pour la maj automatique un script de contrôle sur ce point.

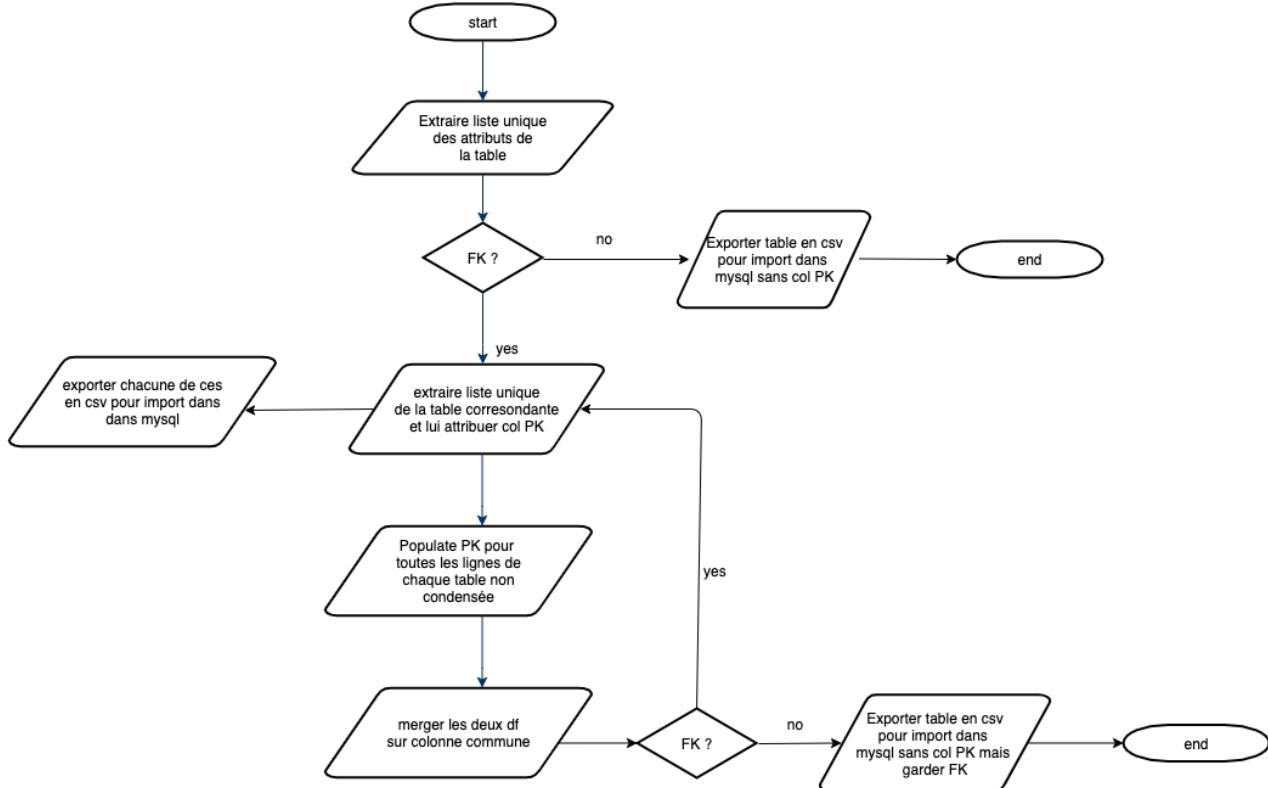
J'a également alerté le producteur de ces données en ligne et a ajouté une alerte en-dessous du lien de téléchargement.

The screenshot shows a data portal interface. On the left, there is a search bar and a main content area titled 'Crimes et délits enregistrés par les services de gendarmerie et de police depuis 2012'. Below the title, it says 'Ce jeu de données provient d'un service public'. A 'NEC MERGITUR' button is present. The main content area contains text about the dataset's purpose and availability. On the right, a modal window titled 'Anomalies' is open, listing two items: 'gendarmerie 2019: mauvaise numérotation des index' (19 juin 2020 12:56) and 'Doublons de gendarmeries en 2017' (9 mars 2018 14:53). At the bottom of the modal are buttons for '+ NOUVELLE ANOMALIE' and 'FERMER'. Below the modal, there is a download section for the dataset 'crimes-et-delits-enregistres-par-les-services-de-gendarmerie-et-de-police-depuis-2012.xlsx', which is 3.4Mo and 543 pages, marked as 'Disponible'. Buttons for 'PRÉVISUALISER' and 'TÉLÉCHARGER' are available, along with a preview icon.

b. Structuration des tables du la base de données avec python/pandas

Une fois vérifié la cohérence des données dans le dataset, on procède à la structure de chaque table de la base de données. Une fois la table structurée sous forme de dataframe, ce dernier est exporté au format 'csv' sans colonne 'clé primaire' (sont toutes auto incrémentée sauf pour département et région).

Processus simplifié de structuration de chaque table dans Python



Exemple de script avec la structuration des tables 'Service' et 'Pointage' pour les données gendarmerie:



```
82]: 1 def CombinaisonUniqueService():
2
3     v=pd.DataFrame({'numDep':[], 
4                      'nomService':[]})
5
6     for sheetname in liste: #liste= liste des noms des onglets
7         if sheetname!="Services GN 2017":
8             # je récupère la liste des nom colonnes (dep-service)
9             dep_serv=pd.read_excel(wbpd,sheetname, header=[0,1], nrow=0, index=0).columns[2::]
10            v=pd.concat([v, pd.DataFrame({'numDep':[i[0] for i in dep_serv],
11                                         'nomService':[i[1] for i in dep_serv]}), axis=0])
12
13     if sheetname=="Services GN 2017": #année où il y a un problème doublons de colonnes
14         dep_serv=pd.read_excel(wbpd,sheetname, header=[0,1], nrow=0, index=0).columns[2::3]
15
16     v=pd.concat([v, pd.DataFrame({'numDep':[i[0] for i in dep_serv],
17                                         'nomService':[i[1] for i in dep_serv]}), axis=0])
18
19     v=v.drop_duplicates(keep='last').reset_index(drop=True)
20
21     return v
```

```
83]: 1 df_combinaisons_uniques=CombinaisonUniqueService()
2 df_combinaisons_uniques.head()
```

```
:
```

	numDep	nomService
0	07	CGD PRIVAS
1	33	CGD BORDEAUX-BASTIDE
2	95	CGD ARGENTEUIL
3	987	CGD ILES-DU-VENT-PAPEETE
4	24	CGD RIBERAC

```
9]: 1 def tableService():
2     df_tableService=df_combinaisons_uniques.copy()
3     df_tableService['idAdm']=2 #gendarmerie PK=2
4     df_tableService['idDirection']="" #pas de directions données pour la gendarmerie (uniquement pour la police)
5
6     return df_tableService
```

```
:0]: 1 df_table_service=tableService()
2 df_table_service.head()
```

	numDep	nomService	idAdm	idDirection
0	07	CGD PRIVAS	2	
1	33	CGD BORDEAUX-BASTIDE	2	
2	95	CGD ARGENTEUIL	2	
3	987	CGD ILES-DU-VENT-PAPEETE	2	
4	24	CGD RIBERAC	2	

```
:1]: 1 len(df_table_service)
```

```
395
```

```
1 # export csv pour import SQL
2 service_to_sql = df_table_service.to_csv('fichiersPourImportSQL/export_services_gendarmerie.csv',
3                                         columns=['numDep',
4                                                   'nomService',
5                                                   'idDirection',
6                                                   'idAdm'], index=False)
```

```
:
```

```
1 #verification que toujours même nombre de services uniques
2 len(df_combinaisons_uniques)
```

Début structure
«Service » en première place
le problème de doublons de colonnes relevé

Début structuration table
«Service » en prenant en compte
le problème de doublon de
colonnes relevé plus haut.

```
[1]: table_service_PG = pd.concat([table_service_police,table_service_gendarmerie])
[2]: table_service_PG.tail()
:
  numDep      nomService  idDirection  idAdm
390      44  CGD ANCENIS ST GEREON      0      2
391      53  CGD CHATEAU GONTIER SUR MAYENNE      0      2
392      91  CGD EVRY COURCOURONNES      0      2
393     987  CGD LES ARCHIPELS PAPEETE      0      2
394     987  CGD LES ILES DU VENT FAAA      0      2
```

assemblage des tables
'Service' gendarmerie et police

04/07/2020 octoplus_DataProcessing_Gendarmerie_DEF

In [422]:

```
def tablePointage():
    # Création template DF final
    df_pointage=pd.DataFrame({
        'codeIndex':[],
        'nomService':[],
        'nbreInfractions':[],
        'annee':[]
    })

    # en raison d'une erreur dans le tableau excel de l'année 2017 (voir verif d
    #oubliens début notebook)
    # je sépare le processing entre 2017 et les autres années
    listanneeclean=[name for name in gendarmerie]
    listanneeclean.remove('Services GN 2017')
    #listanneeclean

    # liste pour accueillir df restructurés par année
    df_pointage_liste = []

    # loop pour aller récupérer dans chaque onglet gendarmerie le nbre
    # d'infractions / service/ code index ss orme de DF
    for name in listanneeclean:
        df_temp = pd.read_excel(wbpd,
                               sheet_name=name,
                               header=[1], #ne retenir que la ligne 'services'
                               # on ne garde que code index et pas libelle + services en lign
                               e
                               # et aggregations nbre d'ifnractions:
                               index_col=0).fillna(0.0, axis=1)
        for i in range(1, len(df_temp.columns)):
            df_temp.iloc[:,i]=df_temp.iloc[:,i].apply(lambda x: int(x))

        df_temp=df_temp.drop('Libellé index \ CGD', axis=1).stack().reset_index
        () .rename(
            columns={'Code index':'codeIndex', 'level_1':'nomService', 0:'nbreIn
            fractions'})
            df_temp['annee']=str(name[-4:]) #récupération année à partir noms onglet
            s et mise en colonne
            df_pointage_liste.append(df_temp)

        #processing pour année 2017
        df_2017 = pd.read_excel(wbpd,
                               sheet_name='Services GN 2017',
                               header=[1], #ne retenir que la ligne 'services'
                               # on ne garde que code index et pas libelle + services en lign
                               e et
                               # aggregations nbre d'ifnractions:
                               index_col=0,
                               usecols=[i for i in range(0, 376)]) #376 dernière colonne avant
                               doublons

        df_2017=df_2017.fillna(0.0, axis=1) #remplacé na values par 0.0 1/2

        for i in range(1, len(df_2017.columns)):
            df_2017.iloc[:,i]=df_2017.iloc[:,i].apply(lambda x: int(x)) #conversion
            float value en int 2/2

        #utilisation stack pour avoir services en lignes et nbre d'infraction en col
        onnes aggégées
```

préparation de la table
'pointage' pour la gendarmerie

Remplacement des valeurs
'Nan' par 0. Les '0' seront
remplacés par une valeur nulle
à l'import des données dans
mySQL.

```
04/07/2020
    octoplus_DataProcessing_Gendarmerie_DEF
    df_2017=df_2017.drop('Libellé index \ CGD', axis=1).stack().reset_index().re
    name(
        columns={'Code index':'codeIndex', 'level_1':'nomService', 0:'nbreIn
    fractions'})
    df_2017['annee']='2017' #récupération année à partir noms onglets et mise en
    colonne
    df_pointage_liste.append(df_2017)

    # concatener tous les df restructurés de chaque année
    df_pointage=pd.concat(df_pointage_liste)

    return df_pointage
```

In [423]:

```
df_table_pointage=tablePointage()
```

In [424]:

```
df_table_pointage[df_table_pointage.duplicated()]
```

Out[424]:

```
codeIndex nomService nbreInfractions annee
```

In [425]:

```
df_table_pointage.tail()
```

Out[425]:

codeIndex	nomService	nbreInfractions	annee
40013	107 CGD ILES DU VENT FAAA	202	2017
40014	107 CGD KONE	42	2017
40015	107 CGD LA FOA	49	2017
40016	107 CGD NOUMEA	129	2017
40017	107 CGD POINDIMIE	16	2017

vérification absence de
doublons

```
: 1 table_pointage_PG = pd.concat([table_pointage_police, table_pointage_gendarmerie])
```

```
: 1 table_pointage_PG.head()
```

codeIndex	nomService	nbreInfractions	annee
0	DIDPAF PREVESSIN 01	0	2012
1	CIAT DE BOURG EN BRESSE	0	2012
2	CIAT D'OYONNAX	0	2012
3	CIAT DE CHATEAU THIERRY	0	2012
4	CIAT DE LAON	0	2012

assemblage tables police/
gendarmerie

```
: 1 table_pointage_PG_def =table_pointage_PG.merge(table_service_PG[['nomService', 'idService', 'idDirection']],
: 2     how='left',
: 3     left_on='nomService',
: 4     right_on='nomService'
: 5 )
```

```
: 1 table_pointage_PG_def.head()
```

codeIndex	nomService	nbreInfractions	annee	idService	idDirection
0	DIDPAF PREVESSIN 01	0	2012	1	1
1	CIAT DE BOURG EN BRESSE	0	2012	2	2
2	CIAT D'OYONNAX	0	2012	3	2
3	CIAT DE CHATEAU THIERRY	0	2012	4	2
4	CIAT DE LAON	0	2012	5	2

ajout des colonnes foreign
key sur la table 'pointage' et
export de la table en csv

```
: 1 table_pointage_PG_def.to_csv('fichiersPourImportSQL/fictif/export_pointage_PG.csv',
: 2     columns=['annee',
: 3     'nbreInfractions',
: 4     'idService',
: 5     'codeIndex'], index=False)
```

III - Création de la base données et optimisation

a.Définition des clefs primaires

La base de données Octoplus est composée de dix tables. Nous allons les créer en fonction de ce qui précisé dans le MPD (voir plus haut).

Les tables suivantes auront une clef primaire auto-incrémentée:

- Administration
- Direction
- Service
- Indicateur
- uniteCompte

Les tables suivante auront une clef primaire pré-définie:

- Region > numéro de région officiel
- Departement > numéro de département officiel
- Demographie > clef composite (anneeRecensement - numDep)
- Pointage > pas de clef primaire > identifiée par les clef primaires des deux tables Service et Infraction dont elle est issue (idService, codeIndex)
- Infraction > code index officiel (1 à 107)

Les scripts de création de chaque table sont réalisés dans un seul fichier 'sql' nommé 'creationTables.sql' et qui est lancé à partir du terminal.

La base de données est créée dans le même fichier avec la commande suivante:

```
>> CREATE DATABASE IF NOT EXISTS octoplus DEFAULT CHARACTER SET utf8mb4 ;  
'utf8mb4' permet de prendre en charge tous les caractères unicode en utilisant un stockage sur 4 octets.
```

b.Exemples de création des tables principales

```

CREATE TABLE Demographie(
    numDep VARCHAR(3) NOT NULL,
    anneeRecensement YEAR NOT NULL,
    nbrePopulation INT NULL,
    densite FLOAT NULL,
    nbreCommunes INT NULL,
    age_15 FLOAT NULL,
    age_15_29 FLOAT NULL,
    age_30_44 FLOAT NULL,
    age_45_59 FLOAT NULL,
    age_60_74 FLOAT NULL,
    age_75 FLOAT NULL,
    age_20 FLOAT NULL,
    partCom_moins_200 FLOAT NULL,
    partPop_moins_200 FLOAT NULL,
    partCom_200_9999 FLOAT NULL,
    partPop_200_9999 FLOAT NULL,
    partCom_10000_plus FLOAT NULL,
    partPop_10000_plus FLOAT NULL,
);

CREATE TABLE Service(
    idService SMALLINT(4) NOT NULL AUTO_INCREMENT,
    nomService VARCHAR(100) NOT NULL,
    numDep VARCHAR(3) NOT NULL,
    idDirection SMALLINT(3) NULL,
    idAdm SMALLINT(3) NULL,
    PRIMARY KEY(idService)
)
ENGINE = InnoDB;

CREATE TABLE Pointage(
    annee YEAR NOT NULL,
    nbreInfractions INT NULL,
    idService SMALLINT(4) NOT NULL,
    codeIndex SMALLINT(5) NOT NULL
)

```

clés primaire composite qui permettra d'identifier chaque nouvelle insertion en fonction de l'année de recensement INSEE et le département.

Avec 'NULL' ou 'NOT NULL' on précise les cas où une valeur peut ou pas accepter des valeurs nulles

Les propriétés des attributs sont systématiquement indiqués avec précision en indiquant si les données importées auront un format alphanumérique, numérique, date et quelle place en mémoire elles occuperont.

IV - Importation des données dans la base

Après avoir créé les tables dans MySql, les données peuvent être importées dans la base de données. J'utilise la commande '**LOAD DATA INFILE**' pour charger les données à partir des fichiers au format CSV préparés dans notebook.

Exemple de chargement de données:

```

LOAD DATA INFILE '/Users/geraldbouget/dumps/export_services_PG.csv'
INTO TABLE Service
FIELDS TERMINATED BY ","
ENCLOSED BY ""
LINES TERMINATED BY "\n"
IGNORE 1 LINES
(numDep, nomService, @value, idAdm)
SET
idDirection = if(@value=0, NULL, @value);

LOAD DATA INFILE '/Users/geraldbouget/dumps/export_pointage_PG.csv'
INTO TABLE Pointage
FIELDS TERMINATED BY ","
ENCLOSED BY ""
LINES TERMINATED BY "\n"
IGNORE 1 LINES
(annee, nbreInfractions, idService, codeIndex);

```

Pour la FK idDirection de la table Service je précise que la valeur sera nulle pour les cas la valeur est 0 dans le fichier csv importé (Pandas export) - pas de

Tous les scripts d'importation sont réalisés dans un fichier unique nommé 'loadData.sql' et est exécuté via le terminal.

V -Ajouts contraintes et optimisation de la base données (backup, automatisation,...)

a. Ajout des contraintes

Une fois les données importées dans la base, je crée les contraintes et notamment j'indique à MySql quelles sont les clés étrangère sur chaque table.

Exemple:

```
ALTER TABLE Pointage
ADD CONSTRAINT FK_idService
FOREIGN KEY (idService) REFERENCES Service(idService)
ON UPDATE CASCADE,
ADD CONSTRAINT FK_codeIndex
FOREIGN KEY (codeIndex) REFERENCES Infraction(codeIndex)
ON UPDATE CASCADE;

ALTER TABLE Infraction
ADD CONSTRAINT FK_idIndic
FOREIGN KEY (idIndic) REFERENCES Indicateur(idIndic)
ON UPDATE CASCADE,
ADD CONSTRAINT FK_idUc
FOREIGN KEY (idUc) REFERENCES uniteCompte(idUc)
ON UPDATE CASCADE;
```

Pour optimiser les tables je précise que la FK se mettra à jour si la PK correspondante est modifiée et assure une cohérence et intégrité des données dans la base.

b. Vérification de la création des tables, insertion des données et ajout des contraintes

À chaque étape je vérifie les modifications ont bien été prises en compte.

J'utilise les commandes suivantes:

```
|mysql> show tables;
+-----+
| Tables_in_octoplus |
+-----+
| Administration
| dash_query_main
| demographie
| Departement
| direction
| Indicateur
| infraction
| Pointage
| Region
| repart_1000hbts
| Service
| uniteCompte
+-----+
12 rows in set (0,00 sec)

mysql> describe pointage;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| annee | year(4) | NO | | NULL | |
| nbreInfractions | int(11) | NO | | NULL | |
| idService | smallint(4) | NO | MUL | NULL | |
| codeIndex | smallint(5) | NO | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0,03 sec)

mysql>
```

```
[mysql> SELECT COUNT(*) FROM POINTAGE;
+-----+
| COUNT(*) |
+-----+
| 885211 |
+-----+
1 row in set (0,36 sec)

[mysql> SELECT
->     TABLE_NAME,
->     COLUMN_NAME,
->     CONSTRAINT_NAME,
->     REFERENCED_TABLE_NAME,
->     REFERENCED_COLUMN_NAME
->   FROM
->     INFORMATION_SCHEMA.KEY_COLUMN_USAGE
->   WHERE
->     REFERENCED_TABLE_SCHEMA = 'octoplus';
+-----+-----+-----+-----+-----+
| TABLE_NAME | COLUMN_NAME | CONSTRAINT_NAME | REFERENCED_TABLE_NAME | REFERENCED_COLUMN_NAME |
+-----+-----+-----+-----+-----+
| demographie | numDep | FK_numDepDemo | departement | numDep
| Departement | idRegion | FK_idRegion | region | idRegion
| direction | idAdm | FK_idAdmDir | administration | idAdm
| infraction | idIndic | FK_idIndic | indicateur | idIndic
| infraction | idUc | FK_idUc | unitecompte | idUc
| Pointage | codeIndex | FK_codeIndex | infraction | codeIndex
| Pointage | idService | FK_idService | service | idService
| Service | idAdm | FK_idAdm | administration | idAdm
| Service | idDirection | FK_idDirection | direction | idDirection
| Service | numDep | FK_numDep | Departement | numDep
+-----+-----+-----+-----+-----+
10 rows in set (0,04 sec)
```

c. Sauvegarde automatisé de la base de données avec CRONTAB

Pour sauvegarder la base de données dans un fichier SQL, j'utilise l'outil 'mysqldump'.

Cet outil sert à écrire un fichier contenant toutes les directives SQL permettant de recréer une base de données à l'identique de l'état dans lequel elle se trouvait au moment de la sauvegarde.

```
(base) MBP-de-Gerald:~ geraldbouget$ cd /usr/local/mysql/bin
(base) MBP-de-Gerald:bin geraldbouget$
(base) MBP-de-Gerald:bin geraldbouget$ ./mysqldump -u root -p --opt --databases octoplus > /Users/geraldbouget/dumps/octoplus_backup_200620.sql
Enter password: [REDACTED]
```

Pour automatiser la sauvegarde, j'utilise la commande bash Crontab:

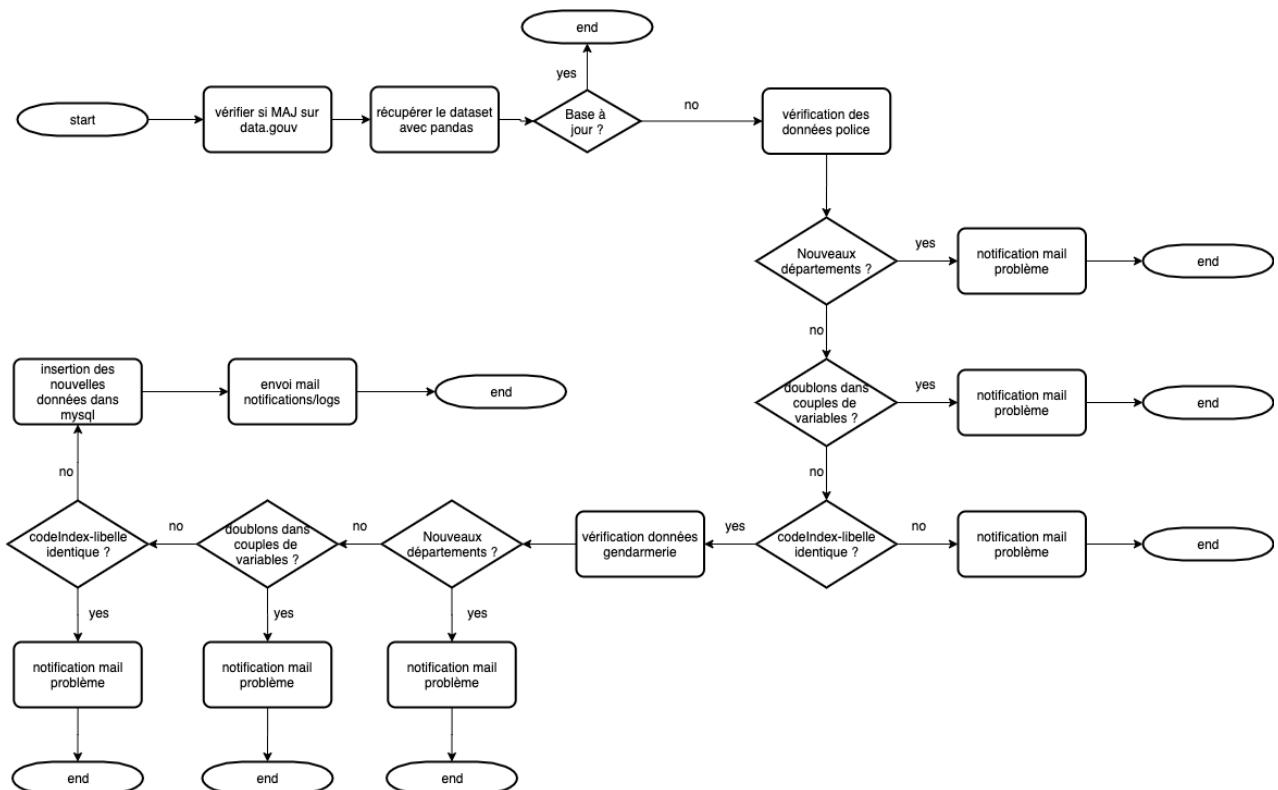
```
(base) MBP-de-Gerald:~ geraldbouget$ crontab -
00 12 28 * * cd /usr/local/mysql/bin;MYSQL_PWD='rootroot' ./mysqldump -u root --opt --databases octoplus > /Users/geraldbouget/dumps/octoplus_backup_$(date +\%Y\%m\%d\%T\%H\%M\%S).sql
```

d. Application python pour mise à jour automatisé de la base de données et de contrôle des données avant import

Afin de mettre à jour la base automatiquement je vais créer une application python qui permettra de réaliser les tâches suivantes:

- vérifier chaque semaine sur le site internet du dataset si une mise à jour a été effectuée
- récupérer en ligne le dataset
- vérifier la cohérence des données du dataset
- insérer les données dans la base de données
- envoyer un mail listant les données insérées dans la bases ou les erreurs rencontrées .

Flowchart application mise à jour automatique de la base de données



Structure du dossier de travail :

```
└── _main.py
└── connexionMySql.py
└── dashQueryUpdate.py
└── data
    ├── gendarmerie_for_update.csv
    └── police_for_update.csv
└── environment.yml
└── gendarmerieUpdate.py
└── issues.py
└── loadingData.py
└── maj_octoplus.log
└── notes_automatisation.rtf
└── onlineUpdateDate.py
└── policeUpdate.py
└── security.py
```

Le projet est composé des classes suivantes:

- classe 'OcotplusUpdate':

chargée principalement de lancer les différents scripts dans un ordre précis et de vérifier si la base a besoin d'être mise à jour après vérification sur le site internet via un script de scraping.

- classe 'ConnexionBase':

connexion à la base de données mySql

- 'loadingData.py':

contient variable du fichier en ligne récupéré avec la commande 'pandas.ExcelFile()'

- 'classe Issues':

récupère tous les message d'erreur ou des données mise à jour et envoie mail avec fichier joint de ces notifications.

- classes 'gendarmerieUpdate' et 'policeUpdate':

vérifie la cohérence et l'intégrité des données à importer et procède à l'insertion des données dans mySql.

- classe 'dashQueryUpdate':

mise à jour automatique de la vue matérialisée qui sert de requête pour l'application dashboard.

L'application est automatiquement exécutée via la commande 'crontab' une fois par semaine.

e. Autres Optimisation de la base de données

- Création d'indexes sur les tables service et infraction

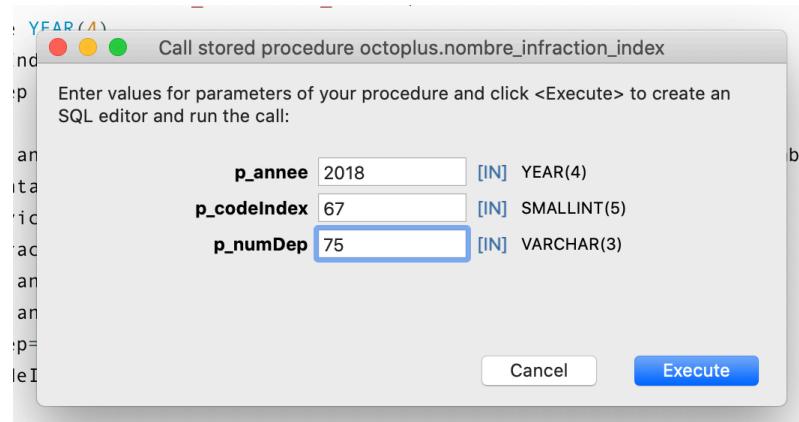
```
mysql> show index from infraction;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| infraction | 0 | PRIMARY | 1 | codeIndex | A | 107 | NULL | NULL | BTREE |  | |
| infraction | 1 | FK_idIndic | 1 | idIndic | A | 13 | NULL | NULL | YES | BTREE |  |
| infraction | 1 | FK_idUc | 1 | idUc | A | 9 | NULL | NULL | YES | BTREE |  |
| infraction | 1 | i_indexLibelle | 1 | codeIndex | A | 107 | NULL | NULL | BTREE |  |
| infraction | 1 | i_indexLibelle | 2 | libelle | A | 107 | 20 | NULL | YES | BTREE |  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0,00 sec)

mysql> show index from service;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Service | 0 | PRIMARY | 1 | idService | A | 1209 | NULL | NULL | BTREE |  | |
| Service | 1 | FK_numDep | 1 | numDep | A | 105 | NULL | NULL | BTREE |  |
| Service | 1 | FK_idDirection | 1 | idDirection | A | 9 | NULL | NULL | YES | BTREE |  |
| Service | 1 | FK_idAdm | 1 | idAdm | A | 2 | NULL | NULL | YES | BTREE |  |
| Service | 1 | i_nomService | 1 | nomService | A | 1209 | NULL | NULL | BTREE |  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0,00 sec)

mysql> 
```

- Création d'une procédure stockée permettant interrogation rapide de la base sur requêtes fréquentes

```
-- PROCEDURE STOCKEE POUR VERIFIER NOMBRE INFRACTION PAR DEP ET PAR ANNEE
DELIMITER |
CREATE PROCEDURE `nombre_infraction_index`(
    p_annee YEAR(4),
    p_codeIndex SMALLINT(5),
    p_numDep VARCHAR(3))
BEGIN
    SELECT p.annee, p.codeIndex, libelle, s.numDep, sum(nbreInfract) as NombreInfract
    FROM pointage as p
    JOIN service AS s ON s.idService=p.idService
    JOIN infraction as i ON i.codeIndex=p.codeIndex
    GROUP BY annee,p.codeIndex, s.numDep, libelle
    HAVING p.annee=p_annee
    AND numDep=p_numDep
    AND p.codeIndex=p_codeIndex; END |
```



- Création d'une vue matérialisée pour accélérer le temps de requête de la base de données à partir du dashboard.

```
-- vue matérialisée pour query à partir de dash
CREATE TABLE query_dash_main
ENGINE = InnoDB
SELECT dp.numDep, dp.nomDep, ic.nomIndicLight, i.libelle, annee, sum(p.nbreInfracti
ons) AS cumulInfraction,
d.nbrePopulation, d.densite, SUM((nbreInfracti
ons/nbrePopulation)*1000) AS InfPour1000
FROM pointage AS p
JOIN service AS s ON p.idService=s.idService
JOIN infraction AS i ON i.codeIndex=p.codeIndex
JOIN indicateur AS ic ON ic.idIndic=i.idIndic
JOIN departement AS dp ON s.numDep=dp.numDep
JOIN demographie AS d ON dp.numDep=d.numDep
GROUP BY dp.numDep, dp.nomDep, ic.nomIndicLight, i.libelle, annee, d.nbrePopulation, d.densite;
```

•Création utilisateur

avec privilèges limités aux requêtes:

```
-- create utilisateur requetes --
CREATE USER 'user'@'localhost' IDENTIFIED BY 'atb225-R';
GRANT SELECT
ON crimesDelits.*
TO 'user'@'localhost';
```

f. Ajout triggers pour gestion d'erreurs d'insertion et d'update

Pour compléter la gestion des erreurs déjà mise en place dans l'application de mise à jour automatique, il me faut ajouter la gestion des erreurs des cas spécifiques suivants:

- un département inséré dans la table service ne peut pas être différent de l'un des département déjà existants dans la table département
- une nouvelle ligne insérée dans la table pointage ne peut avoir un code index > 107
- une nouvelle ligne insérée dans la table pointage et qui correspond aux libelles 'index non utilisé' ne peut avoir une valeur > 0.

Pour gérer ces risques d'erreurs je vais utiliser la commande CREATE TRIGGER.

Le principe est avant toute chose de créer une table 'erreur' dans laquelle je vais spécifiquement inséré des lignes décrivant l'erreur qui pose problème. L'attribut de cette table se verra attribuer un index unique.

Je crée ensuite un trigger pour chaque erreur (BEFORE INSERT et BEFORE UPDATE) et si l'erreur est détectée, l'erreur correspondante est insérée dans la table erreur. Comme chaque ligne erreur est unique, ceci aura pour effet de retourner une erreur indiquant le libellé de l'erreur.

Exemple trigger :

```
-- creation table erreur gestion de certaines
-- erreur d'insertion et d'update
CREATE TABLE erreur (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    defErreur VARCHAR(255) UNIQUE
);

-- insert valeur d'erreur à faire apparaître quand l'insertion
-- ou update pose problème
INSERT INTO erreur(defErreur)
VALUES
    ('Ce numéro de département n\'existe pas dans la table "departement"'),
    ('Le numéro d\'index ne peut pas être supérieur à 107'),
    ('la ligne libelle "index non utilisé" ne peut contenir des valeurs > 0 ');

-- trigger pour éviter insertion d'un codeindex > à 107 UPDATE
DELIMITER |
CREATE TRIGGER before_update_codeIndexPointage BEFORE UPDATE
ON pointage FOR EACH ROW
BEGIN
    IF NEW.codeIndex > 107
    THEN INSERT INTO erreur(defErreur)
    VALUES ('Le numéro d\'index ne peut pas être supérieur à 107');
    END IF;
END |
DELIMITER ;

-- trigger pour éviter que dans ligne 'index non utilisé' soit
-- insérée valeurs différents de 0 ou null - INSERT
DELIMITER |
CREATE TRIGGER before_insert_nbreInfractions BEFORE INSERT
ON pointage FOR EACH ROW
BEGIN
    IF NEW.codeIndex IN (SELECT codeIndex FROM infraction
    WHERE libelle = 'Index non utilisé') AND NEW.NbreInfractions > 0
    THEN INSERT INTO erreur(defErreur)
    VALUES ('la ligne libelle "index non utilisé" ne peut contenir des valeurs > 0 ');
    END IF;
END |
DELIMITER ;
```

PARTIE 3 : EXPLOITATION DE LA BASE DE DONNÉES

I - Le concept

Le principe est de pouvoir proposer une dashboard interactif en ligne.

Ce dernier disposera des fonctionnalités suivantes:

- Une carte interactive de la France avec les départements sous forme de 'choropleths' permettant de visualiser le nombre d'infractions pour une catégorie et une année qui seront elles-mêmes sélectionnées sous forme de menu déroulant.
- Le passage du curseur sur un département fera afficher une bulle d'informations liées à ce territoire (nombre d'infractions, nom du département, nombre d'infraction pour 1000 habitants, densité du département)
- Un graphique indiquant pour la catégorie d'infraction et le département sélectionnés, l'évolution de cette catégorie d'infraction mais aussi des autres depuis 2012. Un clic sur la carte modifie le graphique en conséquence.
- Un 'pie chart' indiquant la répartition des infractions/libellés composant cette catégorie pour le territoire et l'année sélectionnés.
- Un tableau affichant des données socio-économique du département sélectionné. Un menu multi-choix permet de sélectionner les informations désirées.
- Un deuxième onglet permet d'avoir le même genre d'informations sur paris et la région parisienne
- Un troisième onglet proposera des informations plus générales sur le dataset

Il s'agit d'une première version qui est évolutive.

II - Choix technique et méthode adoptée - conception

Ayant utilisé le langage Python pour travailler sur le dataset du projet, mon choix s'est logiquement porté sur la librairie Dash pour réaliser le dashboard.

Cette librairie est gratuite, open source et très performante en terme de rapidité et d'interactions entre les éléments visuels.

Une première étape avait consisté à réaliser des exploitations du dataset sur des notebook pour une première présentation client début juin. J'avais alors choisi la librairie Plotly qui fait partie intégrante de Dash, sachant que j'allais utiliser cette dernière pour le dashboard. Cela m'a permis de me familiariser avec cette librairie.

Pour la visualisation sur carte géographique j'avais dans un premier temps utilisé la librairie Folium qui donne de bons résultat sur notebook mais que je trouvais un peu lente à l'exécution et son implémentation dans Dash n'était pas très esthétique. J'ai donc finalement opté pour Mapbox beaucoup fluide, rapide et qui est le module conseillé par Dash.

J'ai également utilisé Bootstrap pour la partie responsive design que cette librairie apporte et aussi pour l'aspect mise en forme du dashboard.

Sqlalchemy fait évidemment aussi parti des librairies utilisées puisque l'application interroge directement la base de données pour récupérer les données utilisées dans le dashboard.

Pour éviter un temps de requête trop long, j'ai créé une vue matérialisée pour les données les plus volumineuses ce qui m'a permis de gagner un temps d'affichage/d'exécution conséquent.

Pour créer l'application j'ai utilisé l'éditeur de code Atom.

J'ai utilisé la structure de fichier conseillée par Dash pour créer une application multi-page:

Structuring a Multi-Page App

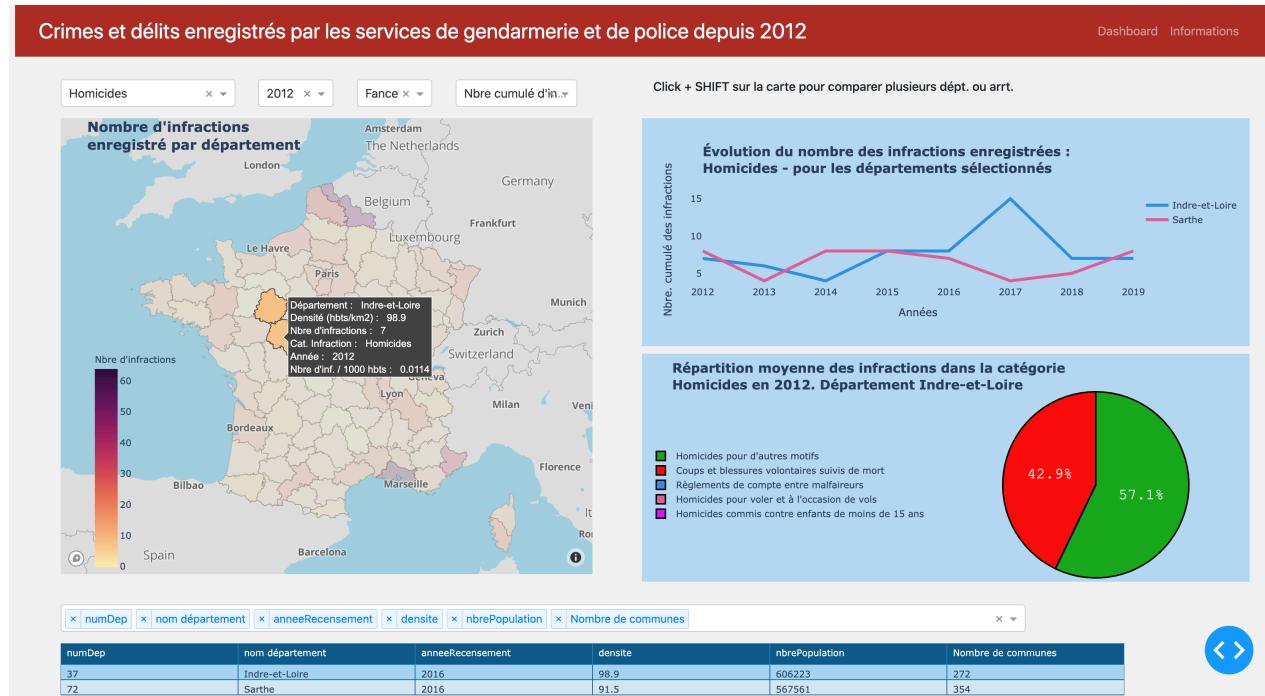
Here's how to structure a multi-page app, where each app is contained in a separate file.

File structure:

```
- app.py
- index.py
- apps
  |-- __init__.py
  |-- app1.py
  |-- app2.py
```

Le fichier ‘__init__.py’ n'est en fait plus indispensable avec Python 3.7 mais je l'ai gardé pour y mettre le script de connexion à ma base données. La connexion se fait donc dès le lancement de l'application.

III - Rendu visuel



Il s'agit d'une première version de l'application tout à fait fonctionnelle mais qui est toujours en cours d'amélioration au moment de la rédaction de ce rapport.

Le script de l'application se trouve en fichier joint.

III - Déploiement du dashboard en ligne

Après validation du bon fonctionnement de l'application en local, j'ai décidé de déployer l'application en ligne.

Ce déploiement implique deux grandes étapes :

- déployer la base de données Octoplus sur un serveur distant
- déployer l'application en ligne

a. Déploiement de la base de données Octoplus sur RDS (AWS)

Le choix s'est porté sur le service RDS (Relational Database Service) de AWS (Amazon Web Service) permet d'installer, de gérer et de mettre à l'échelle facilement une base de données relationnelle dans le cloud et gratuitement pendant un an.

Étapes:

1. Créer un compte sur AWS
2. Créer une instance de base de données Mysql
3. Se connecter à l'instance via le Terminal
4. Créer la base de données Octoplus à partir du backup local

>> Problème de différence de gestion des lettres minuscules entre AWS et mon environnement local : solution >> modification des scripts de création et d'import pour avoir tous les noms de table en minuscule (outils Find/Replace d'Atom)

5. Paramétrer un groupe de sécurité pour permettre à une application externe de se connecter à l'instance.

b. Déploiement de l'application en ligne sur Heroku

Le choix s'est porté sur Heroku qui permet le déploiement d'application de manière assez simple et gratuite.

Étapes :

1. Création d'un compte sur Heroku
2. Installer command line pour Heroku
3. Installer gunicorn
4. Exporter dans un fichier requirements.txt les modules importés dans mon environnement virtuel (conda) >> pip freeze > requirements.txt (modules installés avec pip sur conda) + environnement.yaml
5. Créer un fichier Procfile pour indiquer quel fichier doit être exécuté en premier
6. Téléchargement du fichier de certificat Amazon pour connexion SSL à partir de l'application
7. Upload du fichier certificat sur Heroku:

```
heroku config:set DATABASE_URL="mysql2://admin:<mot de passe>@octoplus-db.*****.amazonaws.com/octoplus?sslca=config/*****.pem" -a octoplus-dash
```

+ commit avec fichier .pem à la racine

8. Script de connexion SSL pour assurer la sécurité des échanges entre l'application et la base de données sur RDS:

```
MYSQL_USER = 'heroku-user'
```

```
MYSQL_PASSWORD = '*****'
MYSQL_HOST_IP = '*****'
MYSQL_DATABASE = 'octoplus'
engine = create_engine('mysql+mysqlconnector://'+ MYSQL_USER +
                      ':' + MYSQL_PASSWORD + '@' + MYSQL_HOST_IP
                      + '/' + MYSQL_DATABASE,
                      connect_args={'auth_plugin': 'mysql_native_password',
                                    'ssl_ca': '****.pem'})
```

9. Création d'un utilisateur spécifique avec uniquement des privilèges 'SELECT' et l'obligation de se connecter en SSL.

```
CREATE USER 'heroku-user'@'%' IDENTIFIED BY 'atb225-R';
GRANT SELECT
ON octoplus.*  
TO 'heroku-user'@'%';
```

10. Se servir de Git pour committer les derniers changement et 'push' sur le serveur Heroku pour exporter l'application :

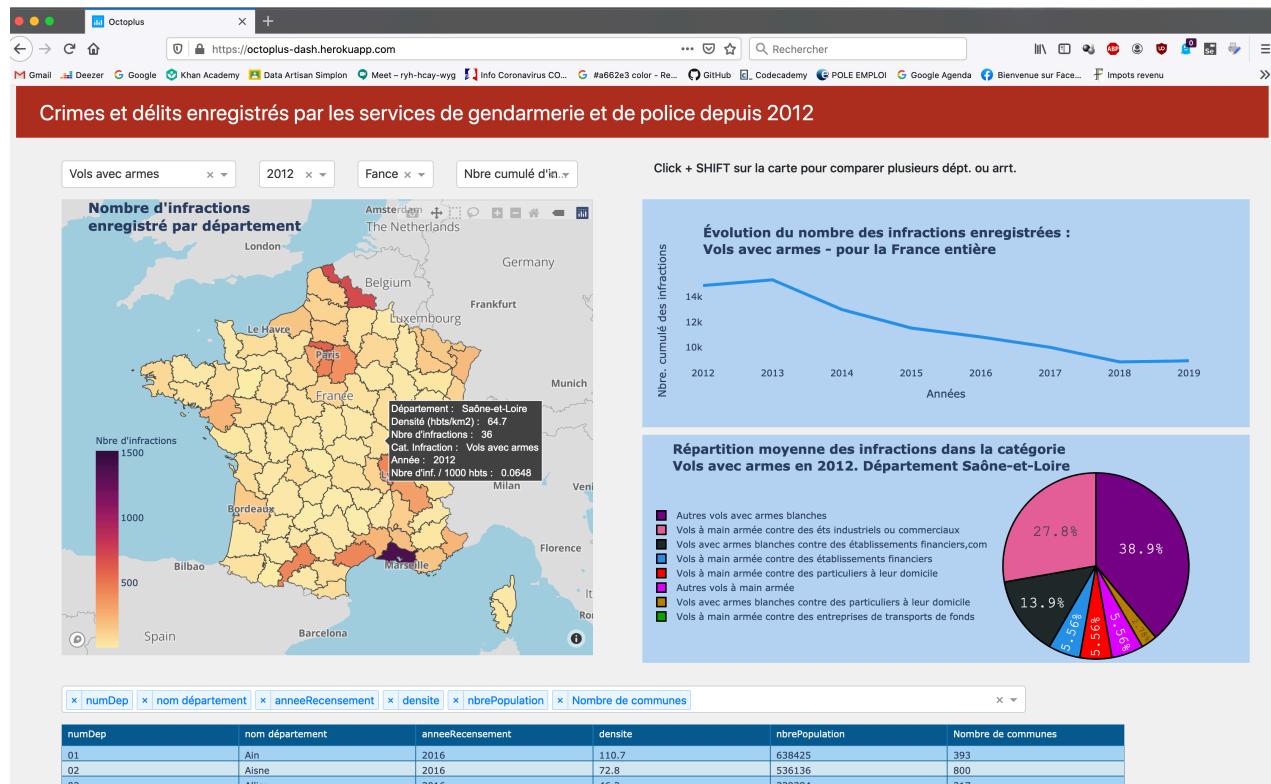
```
$ heroku create my-dash-app # change my-dash-app to a unique name
$ git add . # add all files to git
$ git commit -m 'Initial app boilerplate'
$ git push heroku master # deploy code to heroku
$ heroku ps:scale web=1 #active un dyno (instance)
```

>> problème : la structure des fichiers conseillées par Dash ne convient pas à Heroku

>> solution: exporter l'application temporairement avec une seule page (à ce jour les autres pages peu développées donc Ok - voir plus tard comment structurer les fichiers pour être acceptés par Heroku)

Le dashboard est visible sur :

<https://octoplus-dash.herokuapp.com/>



PARTIE 4 : BILAN ET AMÉLIORATIONS POSSIBLES

J'ai été enthousiasmé de travailler sur ce projet. Non seulement le sujet est très intéressant et m'a permis d'apprendre beaucoup de notions en matière de statistiques sur les crimes et délits, mais j'ai aussi énormément appris et progressé en tant que développeur 'apprenant' tout au long de la préparation de ce projet.

J'ai notamment renforcé mes connaissances et mon expérience en traitement des données en utilisant python. Le développement de la base de données Octoplus m'a permis de mieux comprendre les notions de base de données relationnelles et de conception de ces dernières.

J'ai aussi beaucoup appris en matière de gestion de projet. Mon projet était assez ambitieux avec la création de trois applications. En travaillant à la fois en langage SQL, en Python sur des notebook et sur un éditeur de code (deux de mes applications sont réalisées via Atom), il a fallu s'organiser de manière rigoureuse pour éviter le plus possible des problèmes d'ordre technique, mais aussi en terme de gestion de temps et de tâches.

Trois outils précieux m'ont aidé à rester organisé :

- Trello pour la gestion du temps et des tâches
- Git pour la gestion des versions de code
- Les environnements virtuels pour réaliser mes applications et d'être assuré de travailler dans un environnement stable et identique du début à la fin (conda).

En terme d'amélioration du projet, il me semble que trois axes semblent intéressants:

- Développer la base de données pour ajouter des données externes socio-économiques. Il serait peut être intéressant de prévoir une base de données non relationnelle pour ces données.
- Continuer à améliorer le dashboard en terme d'interactivité, d'esthétique et de rendus visuels.
- Rendre la version déployée multi-page.

CONCLUSION

Travailler sur ce projet a confirmé mes convictions de devenir développeur.

Apprendre à manipuler tous types de données, de les traiter et de les rendre exploitables est passionnant. Chaque étape a ses difficultés et je trouve particulièrement stimulant de devoir résoudre des problèmes à l'aide d'un langage de programmation et de voir ensuite le résultat.

J'ai aussi pris beaucoup de plaisir à développer des applications. Je ne l'ai sans doute pas fait de la manière la plus professionnelle étant donnée mon expérience mais j'ai essayé de m'en approcher et j'ai beaucoup appris. Ça me donne éventuellement envie de me diriger vers cette voie de développeur d'applications tout en continuant à travailler sur les données.

REMERCIEMENTS

Je tiens à remercier l'ensemble de l'équipe encadrante et des formateurs de Simplon qui ont su se rendre disponibles et être force de proposition pour résoudre les difficultés rencontrées.

Je les remercie également pour leur sympathie et leur courage car la période que nous avons traversée entre les grèves et la crise sanitaire, les conditions n'étaient pas propices à travailler sereinement et dans de bonnes conditions. Cependant même à distance les formateurs Yacine et Sayf et notre Cheffe de projet sont toujours restés présents et disponibles.

Je tiens également à remercier mes camarades de promotion avec qui j'ai eu de très bons rapports et j'ai eu plaisir de faire leur connaissance.

BIBLIOGRAPHIE

1. Livres:

- **Statistiques criminelles et enquêtes de victimation**

Auteurs : Alain Bauer, Cyril Rizk, Christophe Soullez

Edition Puf - 2011

- **Modélisation des bases de données**

Auteur : Christian Soutou

Editions Eyrolle - 2017

- **Automate the boring stuff with python**

Auteur : Al Sweigart

Editions No starch press - 2015

2. Sites internet:

- sur le sujet du projet

<https://www.data.gouv.fr/fr/datasets/crimes-et-delits-enregistres-par-les-services-de-gendarmerie-et-de-police-depuis-2012/>

<https://www.interieur.gouv.fr/Interstats/Actualites>

<https://inhesj.fr/>

<https://www.insee.fr/fr/statistiques>

<https://www.observatoire-des-territoires.gouv.fr/outils/cartographie-interactive/#c=home>

- sites techniques

<https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql>

<https://dev.mysql.com/doc/refman/8.0/en/>

<https://dash.plotly.com/>

<https://pandas.pydata.org/pandas-docs/stable/index.html>

<https://git-scm.com/book/en/v2>

<https://stackoverflow.com/>

<https://app.diagrams.net/#>

<https://aws.amazon.com/fr/rds/>

<https://devcenter.heroku.com/categories/reference#deployment>

ANNEXES