

SAE-S5, parcours data : découverte et évaluation d'une technologie de business intelligence

Metabase - DBT



DANNEAUX Lucas – TD1

Année d'étude : 2023-2024

BUT Informatique, IUT Reims-Chalons-Charleville

Enseignants : BLANCHARD FREDERIC | SANDRON LYDIE

Table des matières

Introduction.....	3
Présentation	5
Metabase.....	5
DBT	6
Mise en place d'un Projet d'Utilisation	6
Installation et Configuration.....	6
Installation de Metabase	7
Configuration de Metabase	8
Installation de DBT.....	9
Configuration de dbt.....	9
Présentation Visuelle de Metabase.....	13
Mise en pratique	15
Transformation des données.....	15
DBT	15
Metabase.....	19
Visualisation des données avec Metabase.....	22
Configuration d'options de Metabase.....	26
Paramètres généraux.....	26
Connexion à Slack.....	26
Partage Publique	29
Utilisateurs et Groupe	30
Bases de données.....	30
Métadonnées des tables	31
Autres	31
Avantages, Inconvénients identifiés et Comparaison.....	32
Points Positif	32
Points Négatif	33
Comparaison des outils	35
Transformation de données	35
Visualisation de données.....	35
Conclusion	37
Bibliographie.....	37

Table des Illustrations

Figure 1 : Configuration de la connexion avec une base de données PostgreSQL	9
Figure 2 : Choix de la base de données à utiliser	10
Figure 3 : Contenu du répertoire généré par DBT	11
Figure 4 : Exemple de fichier profiles.yml pour une connexion sur une base PostgreSQL	11
Figure 5 : Configuration du mode de rendu des modèles	12
Figure 6 : Accueil de l'outil Metabase	13
Figure 7 : Liste des options proposé par le bouton Nouveau.....	14
Figure 8 : Modèle du traitement de la table Personne	16
Figure 9 : Exemple du contenu du fichier properties.yml	16
Figure 10 : Rendu dans le terminal de la commande dbt run	17
Figure 11 : Présente des tables des modèles dbt dans Adminer	18
Figure 12 : Rendu dans le terminal de la commande dbt run	18
Figure 13 : Présence en double du département de la Corse-du-Sud.....	19
Figure 14 : Liste des options disponibles.....	20
Figure 15 : Modèle de traitement de la Table Personne	20
Figure 16 : Exemple de données récupéré par le modèle	21
Figure 17 : Aide pour la rédaction du Case sur le Sexe.....	21
Figure 18 : 1ère page du tableau de bord Power BI	22
Figure 19 : Requête de visualisation du nombre de décès par Ville	23
Figure 20 : Exemple de rendu de la requête avec un Graphique en Ligne.....	24
Figure 21 : Exemple de personnalisation disponible pour un Graphique en Ligne ...	24
Figure 22 : Rendu du Graphique en Ligne dans un tableau de bord	25
Figure 23 : Tableau de bord réalisé avec Metabase	25

Figure 24 : Formulaire configuration de Slack	27
Figure 25 : Formulaire de mise en place d'un abonnement Slack	28
Figure 26 : Exemple d'envoi dans Slack	29
Figure 27 : Exemple des ressources générés pour le partage public du tableau de bord du projet	29
Figure 28 : Permission pour le groupe Utilisateur sur les bases de données	30
Figure 30 : Aperçu rapide par Metabase de la table Personne	33
Figure 31 : Formulaire d'ajout de la carte des départements Français	34
Figure 32: Formulaire de connexion à une base de données PostgreSQL de Power BI	36

Introduction

Pour la réalisation de ce projet, j'ai décidé d'étudier le fonctionnement de 2 outils :

- Dans un premier temps, j'ai choisi de me concentrer sur l'outil Metabase qui est un logiciel open-source de réalisation de tableau de bord dont j'ai entendu parler plusieurs fois lors de la réalisation d'entretien pour ma recherche d'alternance, j'ai donc souhaité profiter de ce projet pour découvrir un outil qui me servira dans mon avenir professionnel.
- Dans un second temps, j'ai utilisé l'outil DBT (Data Build Tool) pour réaliser les opérations d'ETL (Extract Transform Load) sur les données avant leurs utilisations dans Metabase afin de pouvoir utiliser cet outil comme un argument de comparaison supplémentaire durant ce projet tout en me permettant la découverte d'un nouvel outil.

Au cours de ce projet, je me suis demandé dans quelle mesure ces nouveaux outils de BI (Business Intelligence) sont-ils efficaces dans un domaine professionnel et qu'elles sont leurs limites d'utilisation.

Je commencerais pour cela par présenter les 2 outils ainsi que leurs principes de fonctionnement, pour ensuite me concentrer sur la manière dont je les ai utilisés dans un contexte de projet professionnel pour les exploiter au maximum pour enfin présenter les différents avantages et inconvénients identifiés ainsi que de réaliser une comparaison avec d'autres outils de BI.

Présentation

Metabase

Metabase est un logiciel de réalisation de tableau de bord de visualisation de données.

Il permet de regrouper des bases de données de différents SGBDR (Système de Gestion de Base de Données Relationnel) afin de pouvoir les analyser ensemble.

Metabase est un outil qui se veut accessible et il présente dans ce but un générateur de requête qui permet aux personnes ne manipulant pas le langage SQL de pouvoir réaliser leurs propres requêtes.

Le logiciel présente une version Open Source (Utilisable Gratuitement) mais il existe des plans proposant des versions de Metabase avec des fonctionnalités plus avancées (Starter, Pro et Entreprise) qui ne seront donc pas exploitées lors de ce projet uniquement réalisé sur la version open source.

Metabase existe sous 2 formes, une version Cloud qui est hébergé chez Metabase mais qui n'est disponible que dans les plans avancés et une version auto-hébergé qui elle est disponible en open source et fonctionne avec le langage Java.

DBT

DBT (Data Build Tool) est un outil de transformation de données qui permet de réaliser les étapes du processus ETL (Extract Load Transform) afin de mettre en place un Data Warehouse (entrepôt de données)

DBT fonctionne à l'aide de modèle écrit avec le langage SQL qui indique avec des scripts les opérations à réaliser sur les données.

Contrairement à Metabase, DBT est un outil qui se veut utiliser par des spécialistes dans le domaine de la DATA qui connaissent le langage SQL ainsi que le fonctionnement d'un Data Warehouse.

L'outil présente également 2 formes, une version Cloud qui est disponible avec les plans avancés ainsi qu'une version Core qui est open source et utilisable avec le langage Python.

Mise en place d'un Projet d'Utilisation

Pour l'utilisation dans un cadre pratique, j'ai décidé de mettre en place la manipulation de ces 2 outils dans le contexte du projet que j'ai réalisé au cours de mon 4ème Semestre en BUT Informatique Spécialité Data à savoir la récupération, la transformation et l'exploitation de données.

Pour cela, j'ai utilisé Metabase à la place du logiciel Power BI afin de réaliser les mêmes visualisations que celle du projet initial. Pour la partie concernant la transformation des données, j'ai manipulé Metabase et DBT à la place des scripts Python afin de pouvoir comparer la difficulté d'utilisation ainsi que l'efficacité des 2 outils. Pour la base de données, j'utilise PostgreSQL à la place de MySQL car ce dernier n'était pas compatible avec DBT.

L'intégralité du projet sera réalisée sur une machine virtuelle utilisant le système d'exploitation Ubuntu dans sa version 22.04 et qui sera vierge de toutes installations.

Installation et Configuration

Installation de Metabase

J'ai tout d'abord commencé par procéder à l'installation du logiciel Metabase sur la machine virtuelle.

Il faut tout d'abord savoir qu'il existe 2 méthodes d'installation de Metabase :

- Par un fichier JAR (Archive de code Java) pour la version Open Source
- Par la création d'une image Docker (Plateforme permettant de lancer des applications à l'aide de conteneur)

Pour la réalisation de ce projet, j'utiliserais la méthode d'installation par le fichier JAR qui nécessite d'avoir le langage Java d'installer sur sa machine

N'ayant jamais installé le langage Java, je me suis tourné vers [la documentation officielle de Metabase](#) concernant l'installation par fichier Jar qui détaillait quelles versions de Java installer pour le fonctionnement de Metabase

Une fois Java installé, il faut procéder au téléchargement du fichier Jar sur le site de Metabase à l'aide de la commande suivante :

wget <https://downloads.metabase.com/v0.47.2/metabase.jar>

Il est possible de télécharger d'autres versions de Metabase en modifiant la valeur de la version dans l'url de la commande (le projet ayant commencé en version 0.47.2 et la version 0.48.1 étant au moment de la rédaction la dernière version sortie)

Pour lancer l'exécution de Metabase, il suffit de lancer la commande suivante

java -jar metabase.jar

Il est cependant conseillé de placer le fichier jar dans un répertoire dédié à Metabase car son exécution va générer de nombreux répertoire et fichiers.

Metabase deviendra alors disponible par default sur le port 3000 de la machine (jusqu'à interruption de l'exécution de la commande)

L'exécution de Metabase entrainera la génération des éléments suivants :

- Répertoire **plugins** : Contient les driver de connexions aux différents services de bases de données supportés par Metabase

- Fichier ***metabase.db.mv*** : Contient les métadonnées ainsi que toutes les informations utilisées par Metabase (requêtes, tableaux, ...)
- Fichier ***metabase.db.trace*** : Contient le journal de débogage de l'outil

Mise à jour

De nouvelle version de Metabase sortant régulièrement, j'ai eu l'habitude de mettre à jour le fichier Jar de Metabase en arrêtant simplement l'exécution de la commande ***java -jar metabase.jar*** afin de remplacer le fichier *metabase.jar* par son homologue le plus récent avant de relancer la commande.

Afin d'éviter les pertes de données avant de mettre à jour Metabase, il est recommandé en cas d'utilisation de la base de données H2 par défaut de mettre une copie du fichier ***metabase.db.mv.db*** en sécurité, ce dernier contenant toutes les tableaux et requêtes de l'outil.

En cas d'utilisation de la version Cloud, les mises à jour sont automatiquement réalisées par les équipes de Metabase.

Configuration de Metabase

Une fois la commande active, il m'a suffi de me connecter sur le port 3000 de la machine afin de pouvoir configurer Metabase avec la langue par default, la création d'un utilisateur Administrateur du Metabase ainsi que l'ajout de la connexion avec la base de données PostgreSQL en remplissant le formulaire affiché une fois le SGBDR sélectionné dans la liste de tous ceux disponible.

PostgreSQL

Afficher le nom

Sae

Host

localhost

Port

5432

Database name

sae5-01

Username

dann0008

Password

.....

Figure 1 : Configuration de la connexion avec une base de données PostgreSQL

Installation de DBT

L'installation de la bibliothèque dbt se fait à l'aide de l'outil pip de python, avant cela, j'ai consulté [la liste des plateformes supportées](#) afin de pouvoir compléter l'installation avec le plugin voulu, il m'a ensuite suffi d'utiliser la commande suivante :

- ***pip install dbt-<plateforme>*** (postgres dans mon cas)

Cette commande installe la bibliothèque de fonctionnement de dbt appelé dbt-core ainsi que la bibliothèque permettant de se connecter à la plateforme de base de données sélectionné

Configuration de dbt

Une fois l'installation de DBT effectué, j'ai procédé à la création du répertoire qui servira à la mise en place de DBT se fait avec la commande suivante :

- ***dbt init nom_repertoire_dbt***

Il m'a alors fallu choisir le type de base de données que je souhaitais utiliser dans ce projet dbt depuis l'invite de commande

```
13:54:26 Setting up your profile.
Which database would you like to use?
[1] postgres

(Don't see the one you want? https://docs.getdbt.com/docs/available-adapters)

Enter a number: 1
```

Figure 2 : Choix de la base de données à utiliser

DBT génère alors un répertoire avec le nom de projet donnée et qui contient les éléments suivants :

- Répertoire **analyses** : Permet de contenir les fichiers **.sql** qui seront compilés mais non exécuté
- Répertoire **macros** : Permet de contenir du code qui pourra être réutilisé dans d'autres script et ainsi éviter la répétition de code redondant
- Répertoire **example** : Contient 2 modèles d'exemples donné par DBT
- Fichier **schema.yml** : Contient les tests sur les schémas de données
- Répertoire **models** : Contient les scripts SQL qui seront exécuter dans le modèle et qui généreront des données selon les modalités décrite dans les scripts et les configurations
- Répertoire **seeds** : Permet de contenir les fichiers **.csv** qui pourront être charger dans la base de données
- Répertoire **snapshots** : Permet d'enregistrer les évolutions des données dans le temps
- Répertoire **tests** : Permet d'exécuter des tests sur les différents modèles de données du projet
- Fichier **dbt_project.yml** : Permet d'identifier le répertoire comme étant un projet DBT avec sa configuration

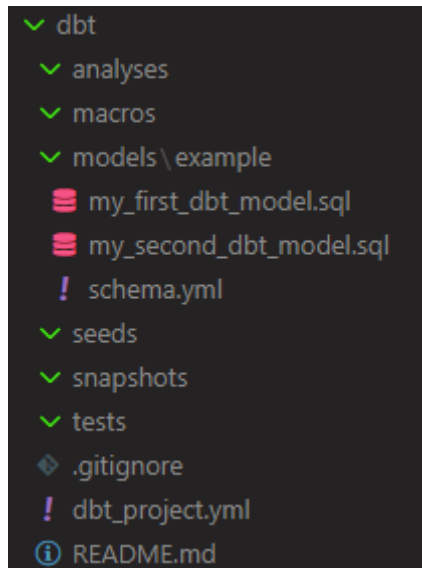


Figure 3 : Contenu du répertoire généré par DBT

Une fois le répertoire généré, il m'a fallu créer un fichier **profiles.yml** dans lequel j'ai configuré la connexion à la base de données PostgreSQL qui est utilisée dans ce projet de mise en pratique

```
sae5_01:
  target: dev # Environnement utilisé par default
  outputs:
    dev:
      type: postgres # Type de base de données utilisé
      host: localhost # nom d'hôte sur lequel se connecter
      user: dann0008 # utilisateur sur lequel se connecter à la base de données
      password: sae51 # mot de passe de l'utilisateur
      port: 5432 # port sur lequel se trouve la base de données
      dbname: sae5-01 # nom de la base de données à utiliser
      schema: public # schéma sur la base de données à consulter
```

Figure 4 : Exemple de fichier profiles.yml pour une connexion sur une base PostgreSQL

Pour vérifier si le fichier contient des informations de connexion valide, il est possible de lancer la commande **dbt debug** qui va tester la connexion à la base de données ainsi que le contenu du fichier **dbt_project.yml** et la validité des dépendances installés.

J'ai ensuite configuré le format de matérialisation de l'exécution des modèles dans le fichier **dbt_project.yml** parmi les 2 possibilités de format à savoir :

- **Table** : Stockera les données dans une table standard

- **View** : Créera une vue avec les données formatées selon les formalités du modèle

```
models:
  sae5_01:
    sae:
      table:
        +materialized: table
      view:
        +materialized: view
```

Figure 5 : Configuration du mode de rendu des modèles

La configuration mise en place effectuera les rendus suivants :

- Les modèles contenu dans le répertoire **table** du répertoire **sae** du projet dbt seront matérialisé comme des tables
- Les modèles contenu dans le répertoire **view** du répertoire **sae** du projet dbt seront matérialisé comme des vues

Présentation Visuelle de Metabase

Lorsque nous accédons à Metabase et que nous nous connectons, nous arrivons sur la page d'accueil de Metabase qui a l'aspect suivant

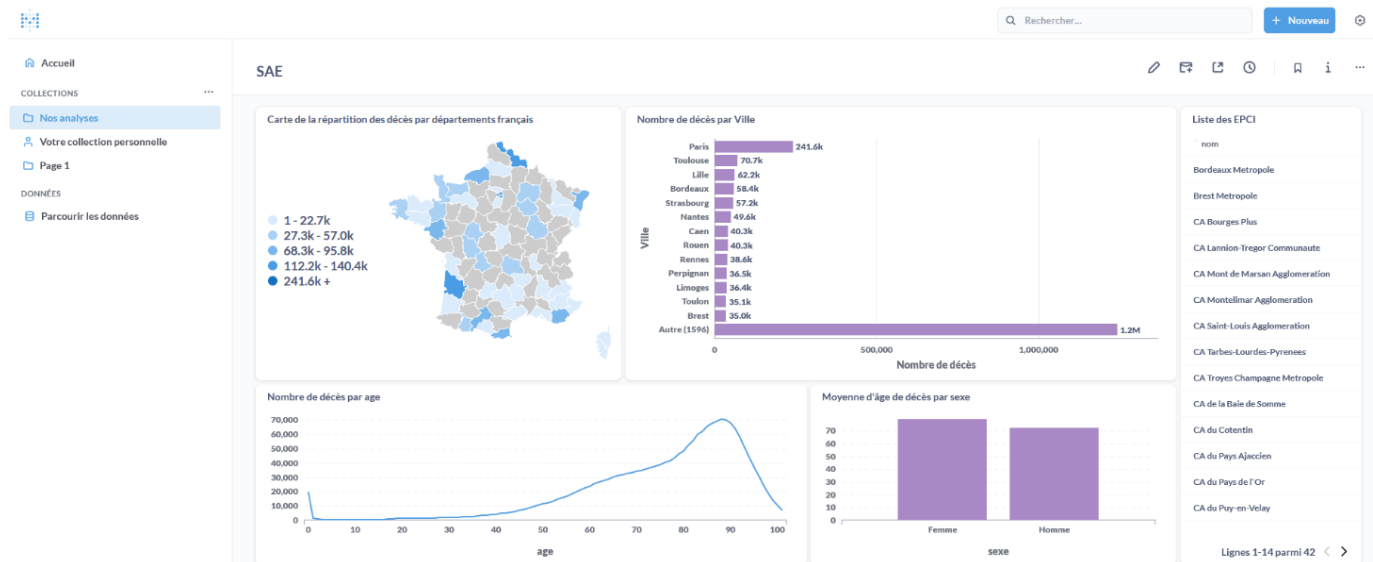


Figure 6 : Accueil de l'outil Metabase

Le menu déroulant à gauche permet d'accéder aux différentes collections de données créées ainsi qu'à toutes les sources de données connectés à l'outil qui nous sont accessibles

En haut à droite de la page se trouve une barre de recherche pour rechercher un élément dans toutes les collections qui nous sont accessibles, les paramètres qui permettent d'accéder aux réglages du compte utilisateur et/ou des réglages administrateur si nous en sommes un.

En haut à droite se trouve également un bouton Nouveau qui permet de créer sur 5 types d'éléments :

- Une **Question** : permet de rédiger une requête de visualisation à l'aide du générateur de requête.
- Une **Requête SQL** : Permet de rédiger une requête de visualisation en utilisant uniquement le langage SQL.
- Un **Tableau de bord** : Un tableau de bord permet de mettre en avant des Visualisation, du Texte ou des Liens vers d'autres éléments

- Une **Collection** : Un répertoire qui pourra contenir des Tableaux, des Requêtes ainsi que des Questions
- Un **Modèle** : Permet de mettre à disposition des ensembles de données plus intuitive pour les utilisateurs à l'aide de questions ou de requêtes

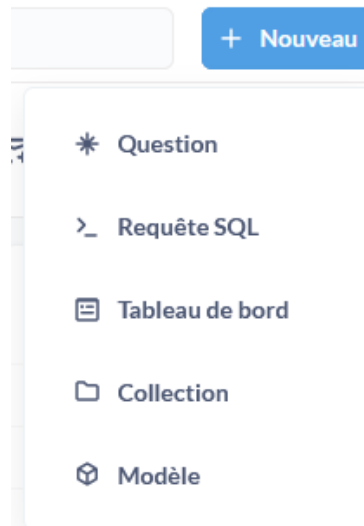


Figure 7 : Liste des options proposé par le bouton Nouveau

Tout le reste de la page correspond à l'affichage d'un tableau de bord qui est défini par l'administrateur et qui n'est donc pas présent en cas de restriction d'accès à ce tableau ou de premier lancement d'un Metabase vierge.

Mise en pratique

Une fois les installations et configurations de Metabase et DBT achevé, j'ai commencé la réalisation de la mise en pratique qui s'est déroulé en 3 étapes.

Dans un premier temps, la réalisation de la transformation des données avec Metabase et DBT.

Dans un deuxième temps, la visualisation des données avec Metabase

Dans un troisième temps, la configuration d'option supplémentaire sur Metabase

Transformation des données

DBT

Pour la réalisation de la transformation de données avec DBT, j'ai commencé par me renseigner sur le fonctionnement des modèles dans DBT à l'aide de [la documentation officielle sur les modèles SQL](#), il est également possible de mettre en place des modèles en Python, cependant, l'intérêt de DBT étant de pouvoir utiliser le langage SQL, je ne me suis concentré que sur l'utilisation de ce dernier.

J'ai alors rédigé 6 modèles à l'aide du langage SQL afin de supprimer les champs et lignes de données inutiles qui était contenu dans la base de données.

Le modèle ci-dessous permet de sélectionner tous les champs id, datedes, age et ville ainsi que le champ sexe qui est modifié pour transformer les booléens **True** or **False** en valeur **Homme** ou **Femme** pour les lignes de la table Personne dont les villes sont dans le rendu du modèle ville et l'âge inférieur à 102.

```

with source_data_personne as (

    select id,
    case
    when sexe = true THEN 'Homme'
    else 'Femme'
    end as "sexe", datedeces, age, ville
    from personne
    where ville in (select id
    |   |   |   |   from "3_ville_traite")
    and age < 102

)

select *
from source_data_personne

```

Figure 8 : Modèle du traitement de la table Personne

Une fois les modèles réalisés, j'ai ensuite créé un fichier properties.yml dans le répertoire des modèles sur DBT selon le modèle présenté dans [la documentation sur la configuration et les propriétés](#). Ce fichier permet de décrire la source de données ainsi que les modèles qui seront créés avec pour la source chaque table ainsi que tous leurs champs alors que pour chaque modèle, seuls les champs qui seront gardés sont à renseigner.

```

sources:
  - name: sae5-01
    description: Base de données contenant les données brutes du projet
    schema: public
    tables:
      - name: ville
        columns:
          - name: id
            description: Clé Primaire de la table
            tests:
              - unique
              - not_null
          - name: nom
            description: Nom de la ville
          - name: code_postal
            description: Code Postal de la ville (inutile donc sera supprimé)

```

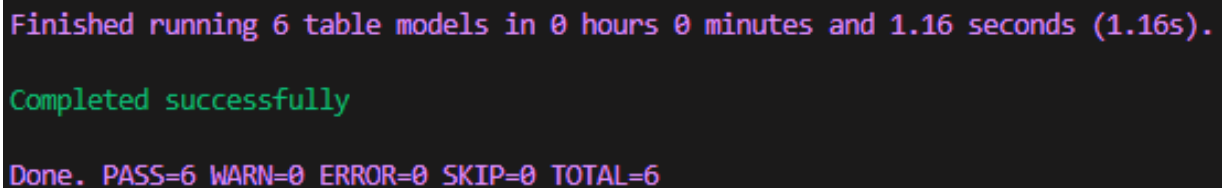
Figure 9 : Exemple du contenu du fichier properties.yml

Il est également possible d'ajouter des tests sur les données afin de vérifier l'intégrité du code SQL rédigé, il est possible de mettre en place des tests dans le fichier `properties.yml` comme visible ci-dessus à l'aide de quatre tests définis par DBT à savoir

- **unique** : Vérifie que la valeur du champ est unique dans la table.
- **not_null** : Vérifie que le champ ne contient jamais de valeur nulle.
- **accepted_values** : Vérifie que la valeur du champ est dans la liste de valeurs possible
- **relationships** : Vérifie que la clef étrangère existe bien dans la table cible.

Il est également possible de rédiger des tests avec des scripts SQL dans le répertoire `tests`, cependant les tests que je souhaitais réaliser sur les données ne nécessitaient pas la rédaction de script.

Une fois la configuration des tests et des modèles terminés, il faut exécuter dans le répertoire de DBT la commande ***dbt run*** qui va exécuter les modèles et ainsi créer les tables ou vues désirées, indiquant dans le terminal si des erreurs sont rencontrées.



```
Finished running 6 table models in 0 hours 0 minutes and 1.16 seconds (1.16s).  
Completed successfully  
Done. PASS=6 WARN=0 ERROR=0 SKIP=0 TOTAL=6
```

Figure 10 : Rendu dans le terminal de la commande dbt run

Il est alors possible de consulter les tables ou vues générées dans une interface graphique de notre SGBDR.

<input type="checkbox"/>	Table
<input type="checkbox"/>	1_region_traite
<input type="checkbox"/>	2_departement_traite
<input type="checkbox"/>	3_ville_traite
<input type="checkbox"/>	4_personne_traite
<input type="checkbox"/>	5_epci_traite
<input type="checkbox"/>	6_conditionmeteo_traite
<input type="checkbox"/>	conditionmeteo
<input type="checkbox"/>	departement
<input type="checkbox"/>	epci
<input type="checkbox"/>	personne
<input type="checkbox"/>	region
<input type="checkbox"/>	ville

Figure 11 : Présente des tables des modèles dbt dans Adminer

Une fois les modèles exécutés, il devient alors possible d'exécuter les tests sur les modèles nouvellement générés afin de tester leurs intégrités avec la commande **dbt test** qui va alors exécuter tous les tests présents dans le répertoire tests et dans le fichier de **properties.yml** regroupant les propriétés des modèles. La réussite ou l'échec des tests sera donné dans le terminal.

La première exécution de **dbt test** visible ci-dessous m'a permis de constater que certains départements avaient le même nom mais des codes différents.

```
Finished running 36 tests in 0 hours 0 minutes and 2.80 seconds (2.80s).
Completed with 1 error and 0 warnings:
Failure in test unique_2_departement_traite_nom (models\sae\properties.yml)
  Got 2 results, configured to fail if != 0
  compiled Code at target\compiled\sae5_01\models\sae\properties.yml\unique_2_departement_traite_nom.sql
Done. PASS=35 WARN=0 ERROR=1 SKIP=0 TOTAL=36
```

Figure 12 : Rendu dans le terminal de la commande dbt run

J'ai ensuite été visualisé le contenu des tables générés sur adminer pour constater la présence en double du département de la Corse-du-Sud avec 2 codes de département différents.

2a	Corse-du-Sud
2A	Corse-du-Sud

Figure 13 : Présence en double du département de la Corse-du-Sud

J'ai ensuite pu corriger l'erreur en modifiant le modèle département pour ne plus sélectionner la région Corse, cette dernière ne représentant que 2 des 2 millions de la table personne.

Metabase

J'ai réalisé la mise en place de la transformation de données dans Metabase bien après celle de DBT car il m'a fallu relire la [Documentation sur les Modèles](#) pour comprendre que leurs fonctions étaient les mêmes que les modèles de DBT.

J'ai réalisé les modèles à l'aide du générateur de requête et des script SQL que j'avais précédemment rédigé pour DBT.

Le générateur de requête demande tout d'abord de choisir les données de départ parmi les différentes bases de données, modèle de données et question disponible. Il est ensuite possible de sélectionner les champs que nous voulons à l'aide du menu déroulant à côté du nom de la source de données.

Il y a ensuite une liste d'option qu'il est possible d'ajouter à la requête :

- Filtre : Permet de filtrer les données à l'aide de condition sur les champs de la source
- Résumer : Permet de mettre en place des métriques ainsi que des agrégations sur les champs de la source
- Joindre des données : Permet de réaliser des jointures avec d'autres sources qui peuvent être n'importe où dans l'application et qui seront ajoutés aux champs de la source
- Colonne personnalisée : Permet d'ajouter une colonne à la requête avec une expression personnalisé

- Trier : Permet de trier le rendu des données selon les champs choisis
- Nombre de lignes maximum : Permet de donner un nombre maximum de ligne à sélectionner

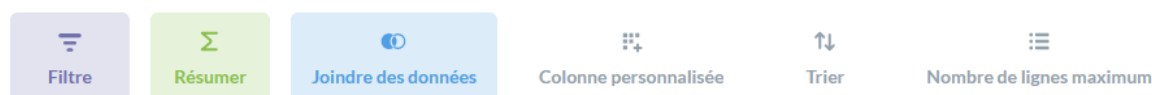


Figure 14 : Liste des options disponibles

Metabase utilisant également le langage SQL pour la réalisation de requête, il m'a suffi de reprendre le principe des modèles DBT en leurs apportant des modifications par l'utilisation du générateur de requête.

Voici par exemple la requête réalisant le modèle sur la table Personne sur le même principe que la dans la Figure 8 : Modèle du traitement de la table Personne. Cette requête commence de la table personne, elle est ensuite jointe au modèle de la table Ville généré précédemment, il est ensuite ajouté la colonne personnalisée sur le Sexe d'une personne et un filtre pour un age inférieur à 102.

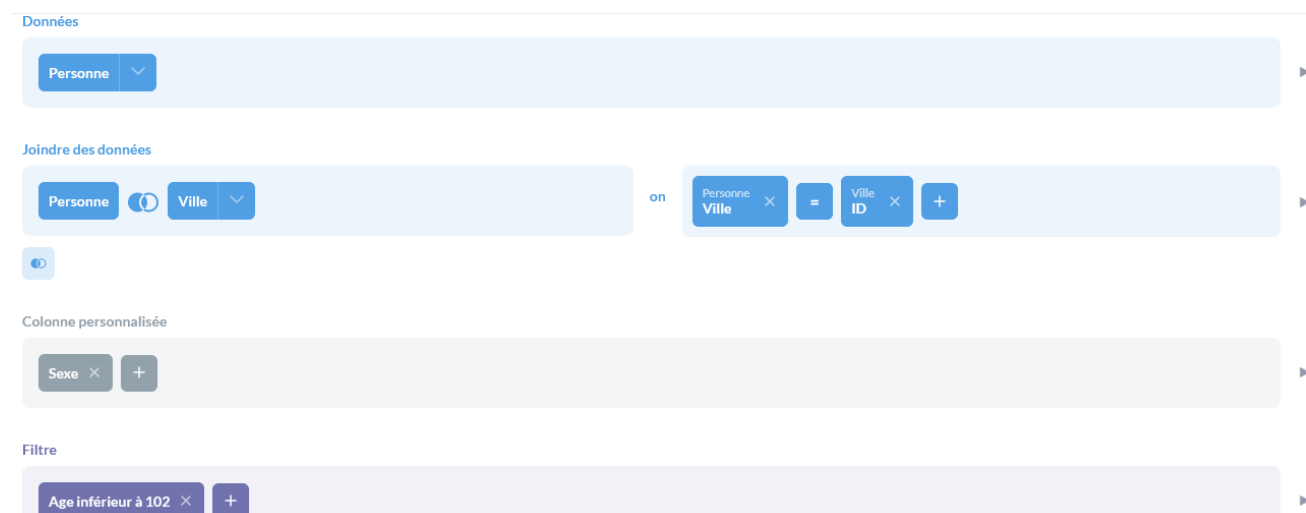


Figure 15 : Modèle de traitement de la Table Personne

Metabase propose en bas de la page un affichage des lignes sélectionné par la requête

ID	Datedeces	Age	Ville	Sexe
442113	février 8, 2013	80	1048	Homme
442114	février 8, 2013	92	940	Femme
442115	février 8, 2013	94	1025	Homme

Figure 16 : Exemple de données récupéré par le modèle

Metabase utilisant une syntaxe différente pour la rédaction des colonnes personnalisés, l'outil affichera automatique une aide avec un exemple pour le type de colonne que l'on souhaite crée.

Voilà ci-dessous l'aide fournie par Metabase pour la création d'une colonne personnalisé de type case pour le sexe d'une personne.

EXPRESSION ⓘ

```
= case([Sexe] = True, "Femme", "Homme")
```

`case(condition, sortie, ...)`

Teste une expression par rapport à une liste de cas et renvoie la valeur correspondante du premier cas correspondant, avec une valeur par défaut facultative si rien d'autre n'est respecté.

condition Quelque chose qui devrait être évalué comme vrai ou faux.

sortie La valeur qui sera renvoyée si la condition précédente est vraie.

... Vous pouvez ajouter plus de conditions à tester.

Exemple

```
case([Poids] > 200, "Large", [Poids] > 150, "Moyen", "Petit")
```

[En savoir plus](#)

Figure 17 : Aide pour la rédaction du Case sur le Sexe

Une fois crée, le modèle peut être enregistré dans la collection de notre choix et devient disponible comme source de données pour la réalisation d'autre requête.

J'ai donc créé les 6 modèles de données avec le générateur de requête sur la base des modèles crée avec DBT.

Une fois les 6 modèles fonctionnels avec les 2 méthodes, j'ai constaté que le contenu des tables étaient identiques avec les 2 méthodes, il est donc désormais possible de manipuler les données de ces modèles.

Visualisation des données avec Metabase

Pour la réalisation des Visualisation avec Metabase, je me suis basé sur les visualisations présentes dans le tableau de bord Power Bi du projet de Semestre 4

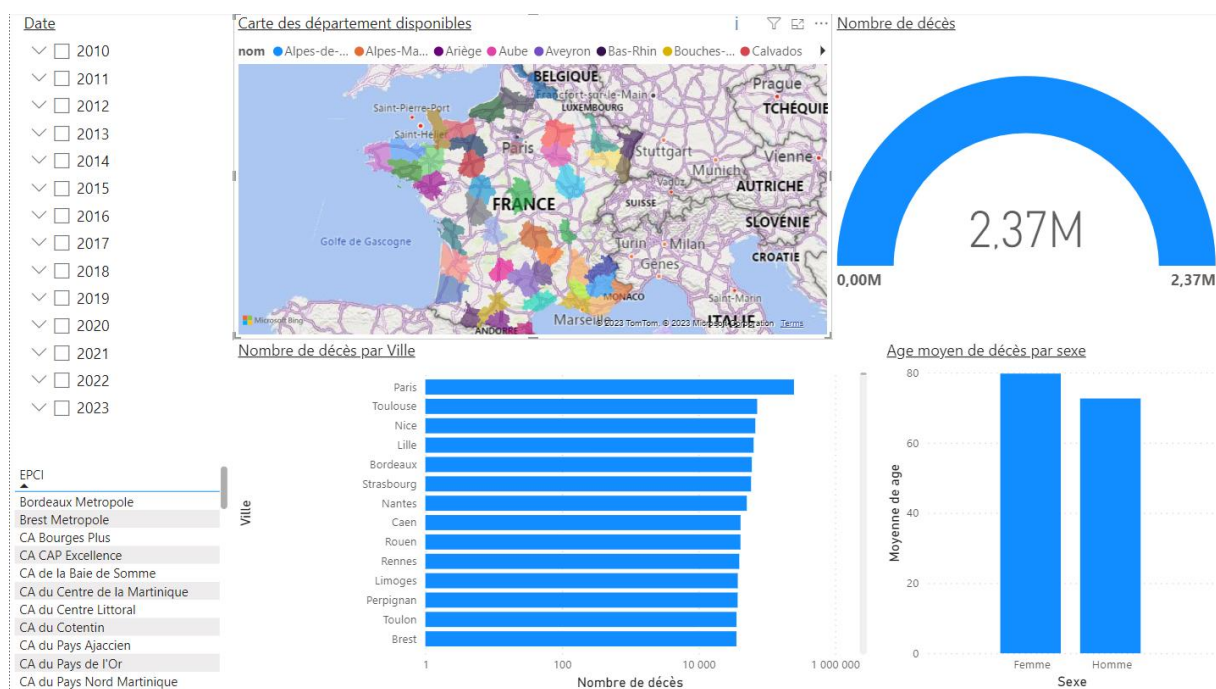


Figure 18 : 1ère page du tableau de bord Power BI

Mon objectif était de réaliser le plus que possible le même type de visualisation en soustrayant les visualisations inutiles afin de pouvoir comparer au mieux le rendu des 2 outils.

Pour réaliser une visualisation sur Power BI, il faut tout d'abord réaliser une Question ou une Requête SQL afin de réaliser la sélection des données que nous souhaitons exploiter pour cette opération.

J'ai personnellement réalisé l'intégralité de mes visualisations à l'aide de question qui me donnait accès au générateur de requête.

Pour chacune de mes requêtes, j'utiliserais les modèles Metabase généré dans la partie transformation de données afin de manipuler des données parfaitement formatées.

Voilà ci-dessous la requête qui sélectionne les villes avec leurs nombres de décès dans l'ordre décroissant.

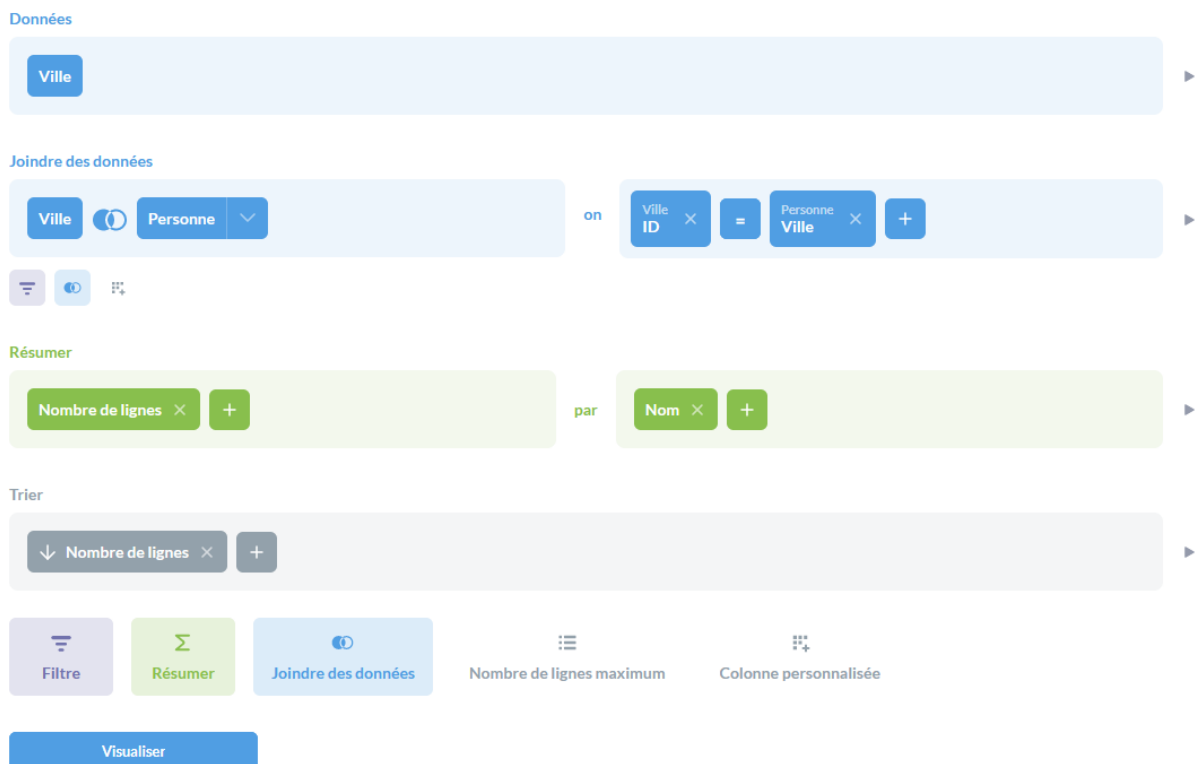


Figure 19 : Requête de visualisation du nombre de décès par Ville

On constate que contrairement au générateur de requête, le bas de la page propose un bouton Visualiser qui permet de passer à la visualisation du rendu de la requête.

Metabase nous présentera alors la liste des visualisations que l'outil permet de réaliser avec son rendu avec les données actuels.

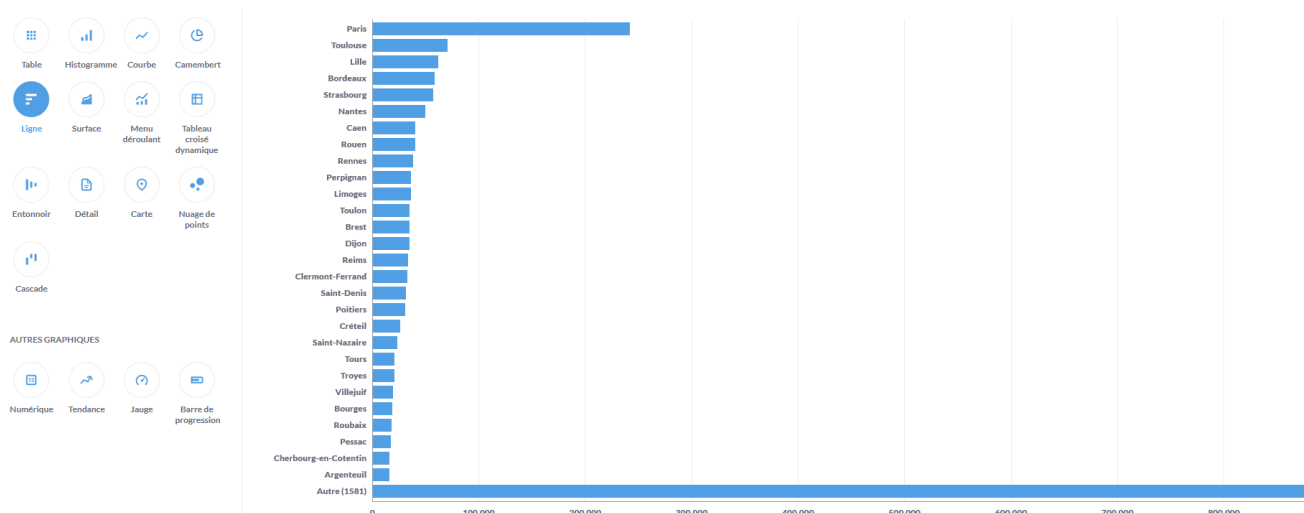


Figure 20 : Exemple de rendu de la requête avec un Graphique en Ligne

Une fois que nous avons trouvé une visualisation qui nous convienne, il est possible de personnaliser le rendu de la visualisation en cliquant sur le bouton option à côté de visualisation

Il est par exemple possible pour un graphique en ligne de modifier la couleur des lignes en abscisses, d'ajouter ou non les valeurs de chaque ligne avec une ligne de but s'il y a une valeur d'objectif à atteindre, il est également possible d'empiler le rendu, enfin, il est possible d'afficher ou non des libellés de légende pour l'abscisse et l'ordonnée pour faciliter la lecture.

Empilement

☒ Ne pas empiler

☐ Empiler (Diagramme à bandes en effectifs)

☐ Empiler - 100% (Diagramme à bandes en fréquence)

Ligne de but ☐

Afficher les valeurs sur les points de données ☒

Formatage des valeurs de libellés

Compacte **Complet**

Figure 21 : Exemple de personnalisation disponible pour un Graphique en Ligne

Une fois la personnalisation terminée il faut enregistrer la question ou la requête SQL dans une collection avant de l'ajouter ou non dans un tableau de bord, il sera alors possible de le positionner dans le tableau de bord choisi.

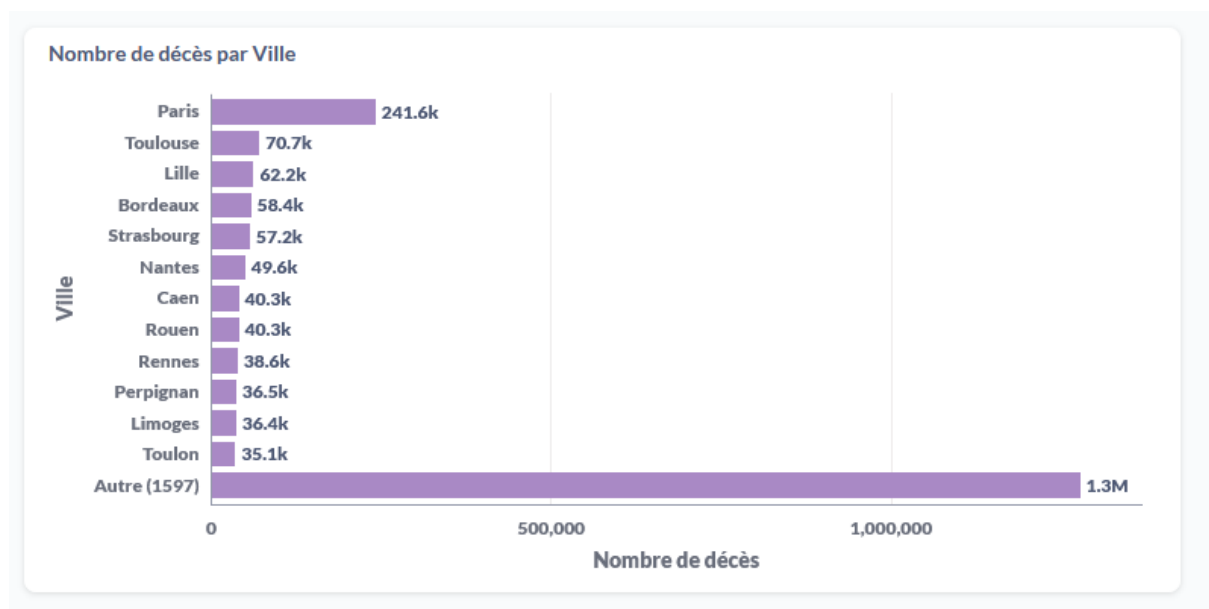


Figure 22 : Rendu du Graphique en Ligne dans un tableau de bord

J'ai réalisé le même procédé pour chacune des visualisations que j'ai réalisé que j'ai placé dans le même tableau de bord pour le rendu final suivant.

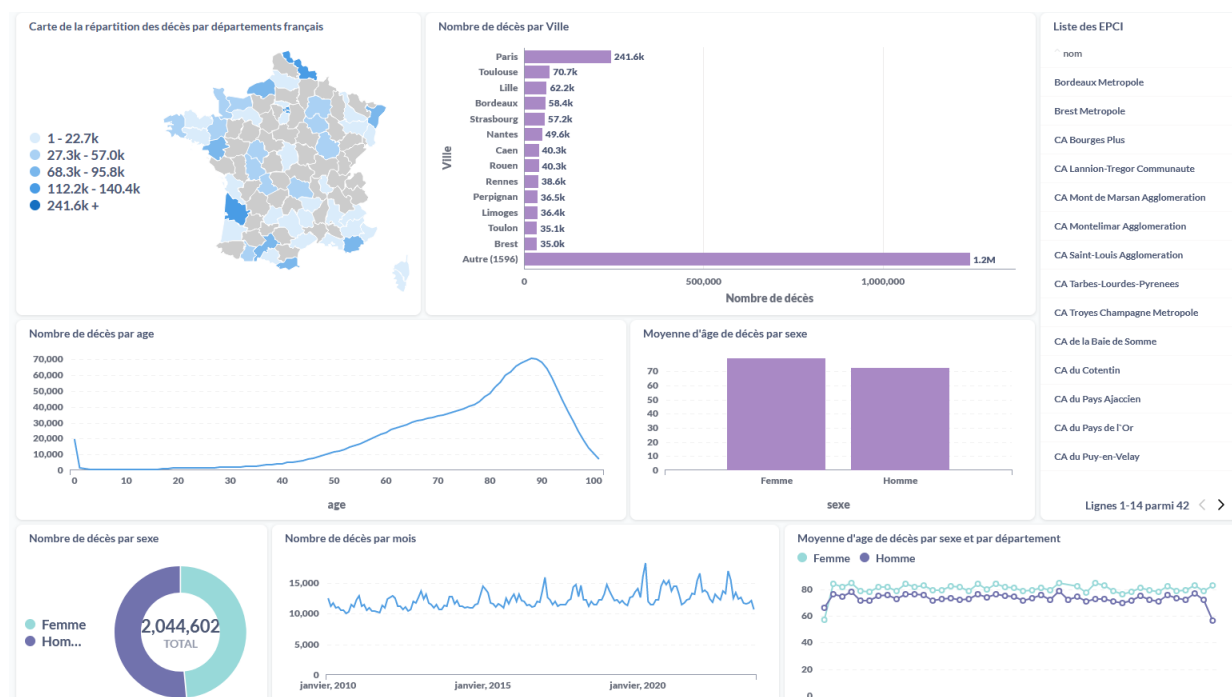


Figure 23 : Tableau de bord réalisé avec Metabase

Une fois le tableau de bord remplis, je me suis occupé de la configuration d'options supplémentaire que propose Metabase.

Configuration d'options de Metabase

Les options administrateurs de Metabase donne la possibilité de mettre en place de nombreuses options.

Je vais détailler dans cette partie les différentes options disponibles et celle que j'ai mise en place.

Paramètres généraux

Il est possible de changer l'url d'accès à l'outil, il sera laissé au port 3000 de la machine pour ne pas perturber le fonctionnement de l'outil.

Un tableau peut être défini comme page d'accueil personnalisé de l'outil, j'ai alors décidé de définir le tableau de bord de la sae comme accueil de mon Metabase.

C'est également dans les paramètres globaux qu'il est possible de décider si les question ou modèle peuvent servir de source de données pour d'autres requêtes, l'utilisation de modèle fessant partie des fonctionnalités que je souhaite tester, j'ai activé cette option.

Connexion à Slack

Metabase permet de mettre en place des abonnements à chaque tableau de bord qui permet de les partager à d'autres utilisateurs automatiquement avec Slack, afin de procéder à la mise en place de cette fonctionnalité, j'ai réalisé la création d'un Slack sae5-01 qui servira de dépôt pour les tableaux de bord généré.

Pour connecter Metabase à Slack il suffit de suivre les étapes données dans l'onglet Slack des paramètres administrateurs qui consiste à créer une application Slack de lui donner l'autorisation de se connecter à Metabase, de récupérer son Jeton d'authentification, de le renseigner sur Metabase et de choisir le canal Slack ou seront stockés les images.

Importez le pouvoir de Metabase dans vos canaux Slack. Suivez ces étapes pour vous connectez à Slack:

1. Cliquez sur le bouton ci-dessous et créer votre Slack App

First, cliquez le bouton ci-dessous pour créer votre Slack App using the Metabase configuration. Once created, click "Installez dans l'espace de travail" to authorize it.

Créer une Slack App

2. Activez le jeton OAuth et créez un nouveau canal Slack.

Cliquez sur "OAuth et Permissions" dans la colonne de gauche, copiez le "Jeton OAuth bot utilisateur" et collez-le ici.

Jeton Oauth de l'utilisateur bot Slack

xoxb-781236542736-2364535789652-GkwFDQoHqzXDVsC6GzqYUypI

Enfin, ouvrez Slack, créer un canal public et mettez son nom ci-dessous. Ce canal ne devrait être utilisé par personne — on y enverra des tables et diagrammes avant d'envoyer les abonnements aux tableaux de bord (c'est un pré-requis Slack).

Canaux publics pour le stockage d'images

metabase_files

Sauvegarder les modifications

Figure 24 : Formulaire configuration de Slack

Un bot utilisateur Metabase rejoindra alors le Slack.

Il est ensuite possible de mettre en place des abonnements sur les tableaux qui permet de configurer :

- La personne ou le canal où envoyer le tableau
- La fréquence de l'envoi (Toutes les Heures, Quotidien, Hebdomadaire, Mensuel)
- Indiquer si l'envoi doit se faire si aucun résultat n'est trouvé sur le tableau

Envoyer ce tableau de bord vers Slack

Les destinataires veront ces données comme vous les voyez, indépendamment de leurs permissions. [En savoir plus.](#)

Post to

Choisissez un utilisateur ou

Envoyé

Toutes les heures ▼

Envoyer vers Slack

Filtrer les valeurs ⓘ

Si un filtre de tableau de bord a une valeur par défaut, elle sera appliquée lors de l'envoi de votre abonnement.

Ne pas envoyer s'il n'y a aucun résultat



Figure 25 : Formulaire de mise en place d'un abonnement Slack

Une fois l'abonnement à Slack réalisé, le bot Metabase a envoyé chaque Jour, chaque visualisation du tableau de bord sous forme d'image dans le Slack que j'utilise pour le projet.

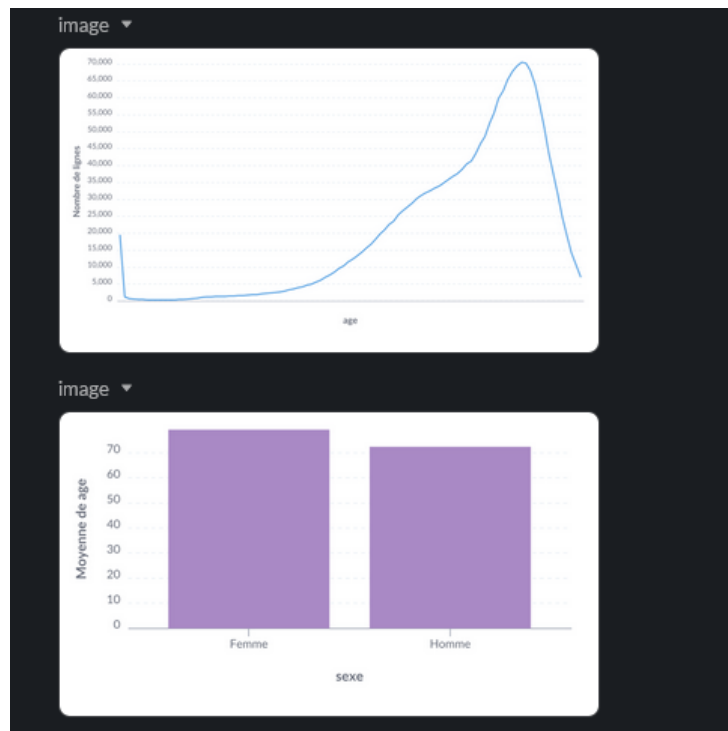


Figure 26 : Exemple d'envoi dans Slack

Partage Public

Il est possible d'activer le partage public des questions et des tableaux de bord.

Un lien ou du code HTML permettant l'intégration dans une page web seront alors générable à partir des éléments pour lesquelles nous souhaitons autoriser le partage public

Partage

Autoriser le partage ☒

Lien public
Partagez dashboard avec des personnes qui ne possèdent pas de compte Metabase en utilisant l'URL ci-dessous :

<http://10.31.32.183:3000/public/dashboard/560ce70a-4df6>

Intégration publique
Intégrez dashboard dans les articles de blog ou les pages web en copiant/collant ce fragment:

```
<iframe src="http://10.31.32.183:3000/public/dashboard/!
```

Figure 27 : Exemple des ressources générées pour le partage public du tableau de bord du projet

Utilisateurs et Groupe

Il est possible pour les administrateurs de Metabase d'inviter des utilisateurs depuis l'onglet Utilisateur en renseignant un formulaire avec le prénom, le nom et l'adresse mail de la personne ainsi que les groupes dans lesquelles on veut ajouter cette personne, l'utilisateur sera ainsi créé. Cette personne recevra alors un mail avec le mot de passe temporaire de son compte Metabase qu'elle pourra changer dans ces paramètres une fois connecté.

Les administrateurs peuvent également créer et gérer des groupes qui leur permettent de restreindre les accès aux données et/ou à la modification des requêtes natives pour chaque base de données connectée au Metabase, il est également possible de restreindre d'autres options comme le téléchargement des données sur les versions avancées de Metabase.

Il existe 3 niveaux d'accès pour les utilisateurs sur une base de données :

- Non restreint : le groupe a libre accès aux données de la base.
- Granulaire : le groupe n'a accès qu'aux tables dont l'utilisateur a autorisé l'accès.
- Pas de libre-service : le groupe n'a pas accès à cette base.

Permission pour le Utilisateur group

1 personne

Rechercher une base de données		
Nom de la base de données	Accès aux données	Modification des requêtes natives
Sample Database	 Pas de libre-service  ▼	 Non 
sae	 Non restreint ▼	 Non  ▼

Figure 28 : Permission pour le groupe Utilisateur sur les bases de données

Bases de données

Les administrateurs ont la possibilité de modifier les connexions avec les bases de données ou de créer de nouvelles connexions avec le même formulaire que celui vu en Figure 1 : Configuration de la connexion avec une base de données PostgreSQL

Métadonnées des tables

Les administrateurs peuvent pour chaque table d'une base de données :

- Décider si cette dernière est interrogeable ou bien masquée aux utilisateurs
- Inspecter le schéma d'origine de la table
- Pour chaque colonne d'une table :
 - Modifier le nom de la colonne
 - Modifier la visibilité de la colonne (colonne visible partout, uniquement dans les vues détaillées, ou alors non inclus (ni visible ni sélectionnable dans le générateur de requête mais toujours dans les requêtes pures))
 - Consulter et/ou modifier le type du champ
- Faire une transformation (pour les champs UNIX vers DateTime)
- Rajouter une description à la colonne

Il est également possible de créer des segments qui servent de filtre sur les données afin de donner accès aux utilisateurs à des données déjà filtrés

Enfin, il est possible de créer des métriques sur une table afin de pouvoir accéder facilement à des valeurs calculées avec des agrégations

Autres

Les administrateurs seront notifiés lors de la sortie d'une nouvelle version de Metabase.

Il est également possible de mettre en place l'envoi de Mail avec un Serveur SMTP permettant d'envoyer des tableaux de bord ou l'invitation d'un utilisateur à rejoindre Metabase.

Il est également possible de mettre en place une Authentification avec un compte Google ou un annuaire LDAP.

Enfin, il est possible d'activer l'intégration de certaines fonctionnalités de Metabase dans des application Node.js ou encore Ruby.

Avantages, Inconvénients identifiés et Comparaison

Points Positif

- L'installation ou la mise à jour de l'outil Metabase est simple et rapide à mettre en place.
- La documentation sur le site de Metabase est très complète et m'a permis de résoudre tous les problèmes rencontrés ainsi que découvrir les différentes fonctionnalités de l'outil.
- L'outil Metabase est doté lors de son installation d'une base de données d'exemple m'ayant servi à réaliser une première manipulation de l'outil avant de réaliser la connexion à la base de données PostgreSQL.
- L'éditeur de requête est un outil qui permet aux personnes ne connaissant pas le langage SQL de tout de même pouvoir réaliser des requêtes.
- Le partage d'un tableau de bord avec des personnes externe à Metabase est facile à réaliser avec la génération automatique d'un lien d'accès du tableau de bord par Metabase.
- Metabase est un outil qui permet de connecter de multiple base de données de différents services facilement
- Metabase est compatible avec beaucoup de SGBDR
- Il est possible de définir des groupes d'utilisateurs afin de leur restreindre les accès aux fonctionnalités voulus
- Il est possible d'envoyer les tableaux par Slack ou par mail avec la configuration d'un serveur SMTP
- Il est possible d'avoir un aperçu rapide de chaque table d'une base de données
- Un utilisateur ne peut personnaliser que la langue d'affichage de Metabase car l'outil souhaite rester le plus simple possible et ne pas perdre les utilisateurs dans un surplus de paramètres.
- Metabase est régulièrement mis à jour

- Metabase propose un Aperçu rapide de chaque table importée

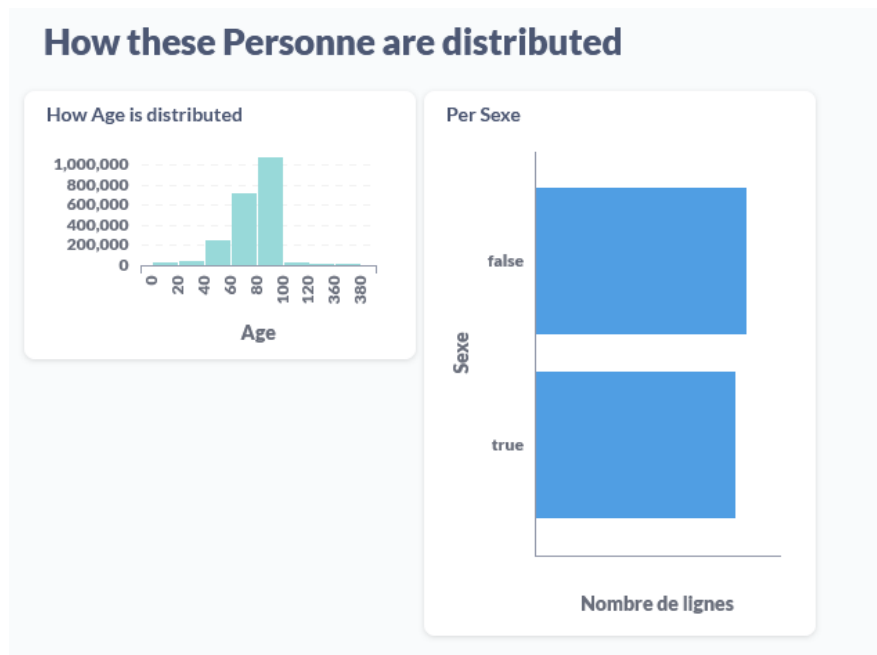


Figure 29 : Aperçu rapide par Metabase de la table Personne

- DBT est un outil simple à installer et à configurer
- Les tests sur les modèles par DBT permettent de facilement détecter des erreurs

Points Négatif

- La documentation sur le site de Metabase bien que très complète n'est actuellement disponible qu'en Anglais ce qui représente un frein dans l'utilisation pour les personnes ne manipulant pas l'anglais contrairement à la documentation de Power BI
- Pour la réalisation d'une visualisation de Carte, il est obligatoire pour un administrateur de procéder à l'import d'une carte GeoJSON à l'aide d'un formulaire si la région concernée par l'affichage n'est pas les Etats-Unis ou une simple carte du monde par Pays

Ajouter une nouvelle carte

Comment voulez-vous appeler cette carte ?

Carte des départements Français

URL du fichier GeoJSON que vous voulez utiliser

geojson.gregoire david.fr/repo/departements.geojson

Rafraîchir

Quelle propriété spécifie l'identifiant de région ?

code

Valeurs d'exemple : 02, 10, 14, 15, 28, 35, 39, 40, 42, ...



Quelle propriété spécifie le libellé de la région ?

nom

Valeurs d'exemple : Aisne, Aube, Calvados, Cantal, E...



Figure 30 : Formulaire d'ajout de la carte des départements Français

- La sélection d'un élément sur une visualisation dans un tableau de bord ne procède pas à sa propagation comme paramètre sur toutes les autres visualisations du tableau
- Metabase devient lent quand utilisé par plusieurs utilisateurs en même temps
- Les graphiques à barres ne sont pas défilable et ne présente donc que les premières valeurs qui correspondent à la requête utilisée
- La documentation de DBT n'est pas très bien organisée et est donc compliqué à comprendre alors que l'outil n'est pas dur à mettre en exécution
- DBT n'est pas compatible avec beaucoup

Comparaison des outils

Transformation de données

Les 2 outils sont assez simples d'utilisation pour des personnes comprenant le principe d'un ETL cependant Metabase permet à des personnes moins expérimentées de mettre en place des modèles grâce aux générateurs de requête.

La possibilité de mettre en place des tests sur les modèles avec DBT permet de détecter des erreurs dans les données sélectionnées là où il n'est pas possible de détecter facilement ce type d'erreur avec les modèles Metabase.

Dans le cas d'une utilisation de Metabase, l'utilisation des modèles de DBT vient rajouter une couche supplémentaire qui n'est pas vraiment justifiée vu que Metabase possède également la possibilité de réaliser des opérations d'ETL.

Contrairement à Metabase, DBT n'est pas compatible avec beaucoup de services de base de données ce qui m'a forcé à modifier mon projet initial pour que celui-ci fonctionne avec PostgreSQL au lieu de MySQL dans la version initiale.

Visualisation de données

Power BI est un outil qui est bien plus complet que Metabase aux niveaux des fonctionnalités cependant cela a un énorme impact sur l'accessibilité du logiciel qui est bien plus difficile à prendre en main pour de nouveaux utilisateurs contrairement à Metabase par son développement tourné pour l'accessibilité.

La mise en place d'une connexion avec une base de données par Power BI est plus difficile car les formulaires sont moins détaillés que dans Metabase qui détaille chaque paramètre à remplir.

Figure 31: Formulaire de connexion à une base de données PostgreSQL de Power BI

Contrairement à Power BI, les tableaux de bord Metabase ne sont pas limités en tailles et permettent donc de contenir autant de Visualisation que voulu là ou Power Bi demande la création de plusieurs pages sur un même tableau de bord, cependant mettre beaucoup de visualisation sur un même tableau de bord rends les requêtes beaucoup plus long à se réaliser à cause de leur nombre là auquel il faut aussi ajouter la possibilité d'autre utilisateurs sur d'autres tableau de bord qui augmente encore la durée des requêtes, là ou Power Bi a prévu une taille limite à une page pour limiter le nombre de requête sur une seule page.

Sur Power BI, la sélection d'un élément le sélectionne comme paramètre qui va alors s'appliquer sur toutes les visualisations de la page et va permettre une analyse plus intuitive des données, Metabase ne propose pas cette option, ce qui rends plus difficile les analyser car sélectionner un paramètre demanderais de modifier chaque requête.

Il y a beaucoup moins de possibilité sur les visualisations dans Metabase par exemple, il n'est pas possible de faire des Treemap ou de faire défiler les graphiques à barres ce qui n'affiche que les premières valeurs contrairement à Power BI qui permet de défiler les graphiques afin de pouvoir visualiser chaque ligne.

Il n'y a pas besoin d'ajouter de fichier GeoJSON pour réaliser des affichages de cartes, Microsoft ayant déjà toutes les cartes disponibles grâce à Bing

Conclusion

Metabase est plus un outil tourné vers pour la plus grande accessibilité pour un suivi simple des données contrairement à Power BI qui correspond plus à un suivi de données complexe avec beaucoup de variabilité.

Pour la réalisation des opérations d'ETL sur des données, les outils Metabase et DBT sont tout 2 très performant dans cette épreuve cependant l'utilisation de DBT avant d'envoyer les données sur Metabase ne présentais pas de grand intérêt.

Dans le cadre professionnel je recommande l'utilisation de DBT et de Metabase mais pas des 2 éléments en même temps.

Bibliographie

[Site des fichiers GeoJSON de la France](#)

[Documentation de Power Bi par Microsoft/](#)

[Documentation de Metabase](#)

[Documentation d'installation de DBT](#)

[Documentation de la création de projet DBT](#)