Qualité des données

Sébastien AMALLA

Objectifs de ce cours

- Dataset donné
- Audit complet de la qualité des données
- Mise en place de scripts de traitement des données
- Automatisation d'une pipeline
- Notion de gouvernance des données
- Exploitation des données
- Réconciliation des données
- Projet sur un autre dataset

Cycle de vie des données

- Création ou collecte
- Stockage
  - BDD, datalakes, clouds, serveurs locaux,...
- Traitement
  - Mise en forme des données, calculs de données secondaires
- Utilisation
  - Analyse, consultation, statistiques, graphes, ...
- Partage / diffusion
  - Droits d'accès, autres systèmes, utilisateurs, API, ...
- Archivage
  - Données mois utilisées mais pas à détruire
- Destruction / anonymisation
  - Suppression lorsque plus utiles, anonymisées pour des stats, respect des RGPD

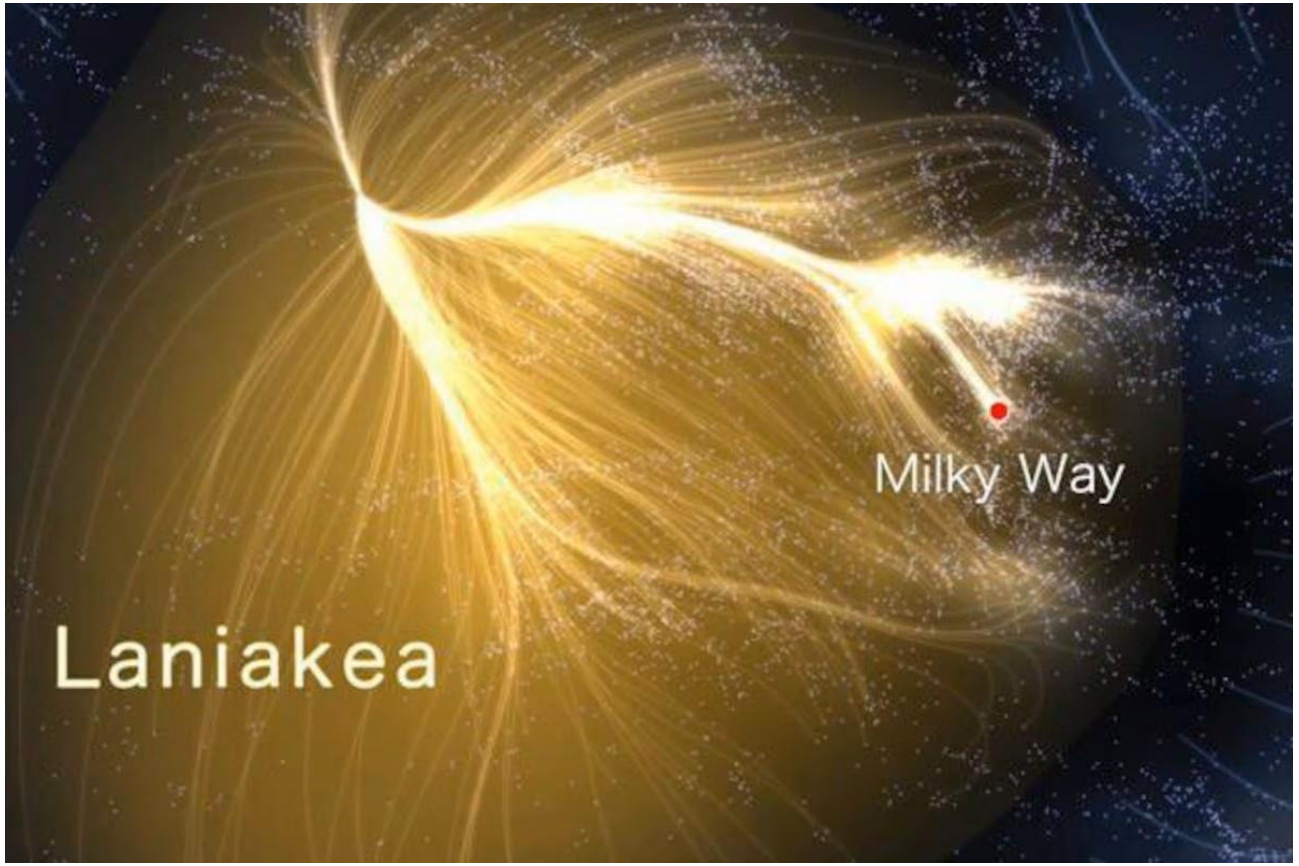
Problématiques liées à la qualité des données

- Exactitude / fiabilité
  - Typos, valeurs incorrectes (NaN ou valeurs aberrantes), doublons, ...
- Complétude
  - Valeurs manquantes pour certaines entrées / entrées manquantes
- Unicité
  - Une donnée devant être unique n'est pas stockée plusieurs fois
- Actualité
  - Données à jour (adresse changée suite à un déménagement par ex)
- Standardisation / format
  - Permet une exploitation fluide si format unifié
- Traçabilité et gouvernance
  - Data lineage : origine et transformations subies
- Sécurité et conformité
  - Anonymisation, respect des RGPD

Problématiques liées à la qualité des données

- Le processus de qualité des données permet donc d'éviter:
  - Des incohérences
  - Une incomplétude
  - Une obsolescence
  - Des formats hétérogènes
  - Un manque de traçabilité
  - Le non-respect des normes (RGPD)
- Si des données ont une qualité insuffisante leur utilisation est compromise, le machine learning notamment
- Impact financier

Découverte de Laniakea



- Exploitation mathématiques de données mesurées par des dizaines d'équipes à travers le monde
- Chaque équipe avait son format
- Chaque équipe avait ses appareils de mesure
- Toutes les équipes ne les ont pas vérifiées
- Travail immense de qualité des données
- A permis une découverte majeure

Plan du cours

- Passage rapide sur la collecte et le stockage des données
  - Modules à part
- Data profiling
- Objectifs et exigences de qualité
- Data cleansing
- Tests automatiques de qualité
- Création d'une pipeline qualité
- Projet de groupe

Collecte et stockage des données

Collecte des données

- Comment collecter des données ?
- Données quantitatives
  - Mesures : de capteur, d'observations
  - Résultats de simulations
  - Enquêtes, questionnaires, articles, …
- Données qualitatives
  - Questions libres sur un sujet
  - Analyses documentaires : rapports, archives, publications, …
  - Recueillir des témoignages
- Collecte manuelle ou automatiques (web scraping)

Collecte des données

- Trois types de données
- Données primaires : enquêtes, capteurs, observations, expériences
- Données secondaires : déjà existantes (bases publiques, archives, publications scientifiques, ...)
- Données générées : Calculées à partir des données primaires et secondaires (déduire un âge depuis une date de naissance, résultats d'une simulation, d'un calcul, ...)
- Quantitatives : Sondages, capteurs, mesures, résultats de calculs
  - Type de données : numériques (valeurs)
- Qualitatives (descriptive, pas de valeur associée) : Contextes, émotions, comportements,
  - Type de données : textes, enregistrements audio/vidéo

Stocker des données

- Différence entre **infrastructure** et **organisation** des données
- Infrastructure : Local ? distribué ? en cloud ?
- Organisation : SQL ? NoSQL ? DataLake ?
- Quelle est la différence entre SQL et NoSQL ?
- Ce sont deux modèles de bases de données répondant à différents besoins
  - Contrairement à ce que certains pensent, NoSQL est bien une BDD

SQL

- Table relationnelle
  - Ligne, colonnes, clé primaire, clé étrangère (liant plusieurs tables)
  - Exemple, une table étudiant, une table cours, une table inscription liant les deux
- Schéma rigide (on-write)
  - Avant d'enregistrer des données, il faut qu'elles respectent le format défini
  - Le schéma est appliqué à l'écriture des données (on-write), contrairement à on-read
  - Contrôle strict des données, très rigide
- Scalabilité verticale
  - Hébergée sur un seul serveur, on augmente le stockage et la ram de celui-ci
  - Limites physiques, single point of failure, moins adapté au big data (milliers de To)
- Cohérence ACID
  - Vérification des transactions méthode ACID : grande sécurité

NoSQL

- Table non-relationnelle
  - Les colonnes ne sont pas les mêmes pour chaque ligne de la table. Chaque ligne a une clé
- Schéma flexible (on-read)
  - Aucun contrôle à l'enregistrement des données
  - Le schéma est appliqué à la lecture des données : flexible, évolutif.
- Scalabilité horizontale
  - Répartition des données sur plusieurs nœuds
  - Ajout de nouveaux nœuds au besoin. Idéal pour du big data.
- Cohérence BASE
  - Transactions plus flexibles avec cohérence éventuelle

Objectifs et exigences de qualité

Exigences de qualité

- La qualité des données dépend du contexte
- Des données considérées de "haute qualité" pour un objectif peuvent être de "basse qualité pour un autre
- Les standards de qualité sont établis par un "data governance council"
- La qualité des données n'est pas un "projet" à réaliser une fois, c'est une politique constante

Exigences sur la complétude

- Imaginons que vous vouliez numériser une bibliothèque.
- Quand est-ce que vous considérerez les données complètes ?
- Tous les livres ont été scannés
- Tous les scan ont été vérifiés
- Tout est bien indexé
- Autre exemple : relevés de capteurs toutes les heures. Comment évaluer la complétude des données ?
- Chaque relevé est complet, il ne manque aucune heure

Exigences sur l'unicité

- Quelles exigences d'unicité dans les cas précédents ?
- Chaque livre n'est enregistré qu'une fois
- Chaque jour n'est enregistré qu'une fois
- Autre exemples :
  - Numéro de client
  - Numéro de sécurité sociale
  - Numéro d'un dossier
- Si une entrée est en doublon et avec des colonnes différentes, c'est un problème majeur à adresser

Exigences sur l'actualité

- Les données doivent être le plus à jour possible
- Représenter la réalité actuelle
- Être disponibles rapidement
- Exemple : prédire l'évolution d'un ouragan
- Les scientifiques ont besoin rapidement de données les plus à jour possible
- Imaginez que les seules données disponibles soient celles d'il y a 3 jours
- Mise à jour en fonction de la fréquence de changement
  - Gardez les données météo du mois dernier n'est pas utile

Exigences sur la fiabilité

- Données cohérentes dans un intervalle donné
- Pas de NaN, pas de -1000°C en température, d'âge négatif
- Correct type de données : des float pour la température, pas de chaînes de caractères
- Même format de données:
  - Dates en DD/MM/YYYY ou DD-MM-YYYY ou MM/DD/YY, … ?
  - Une colonne nom-prenom ou une colonne nom et une colonne prénom ?
- Décrit précisément et fidèlement la réalité
  - Correctes, fiables et vérifiables
- Par exemple : BDD des accidents de la route survenus en Inde, une entrée pour un accident à Paris

Autres questions à se poser :

- **Usage** : Est-ce que les données sont facilement compréhensibles ? Accessibles ? Sont-elles pertinentes ? Peut-on enlever celles qui le sont moins ?
- **Stockage** : Est-ce que l'espace serveur est suffisant ? La BDD va grossir à quel rythme ? La latence est-elle trop grande ?
- **Flexibilité** : Simples à manipuler ? Faciles à comparer ?
- **Sécurité** : Données anonymisée ? BDD bien sécurisée ? RGPD ?
- **Valeur** : Est-ce rentable de maintenir ces données ?

- *Data governance council* : Détermine les aspects critiques de qualité des données et les procédure à suivre pour s'en assurer (KPI)

Quand intervient la qualité des données ?

- Tout au long du cycle de vie des données !
- **Création / Collecte** : Vérification des sources, contrôle de la saisie, validation du format
- **Stockage** : Intégrité et cohérence des données de la base, sécurité
- **Traitement** : Nettoyage, standardisation, cohérence après transformation
- **Utilisation** : Pertinence, complétude, fraîcheur
- **Diffusion** : conformité et lisibilité
- **Archivage** : Pertinence des données, qualité minimale d'archivage

Quelles technologies open source ?

- **Profiling et nettoyage :**
  - OpenRefine (Profiling et visualisation, limité en nettoyage, low-code)
  - Pandas avec pandas-profiling (Python, limité en visualisation)
- **Contrôle qualité :** Complétude, validité, unicité, cohérence, fraîcheur:
  - Soda-core (mesures des KPI, rapports JSON, mais ni nettoyage ni correction).
- **Monitoring :** Affichage graphique et envoie d'alertes
  - Grafana
- **Orchestrateur :** Apache Airflow, Dagster

Profilage

Setup local pour ce cours

- Installer un python 3.10 en local et l'utiliser pour créer le venv
  - Télécharger python 3,10 embedded
  - ouvrir powershell
  - se mettre dans le dossier du projet
  - (path to python.exe) –m venv venv
  - ./venv/Scripts/activate
  - python -V
  - pip install –r requirements.txt
- Téléchargez OpenRefine

Par où commencer ?

- Comprendre le dataset
- Lire les métadonnées et toute la documentation disponible
- Prendre connaissances des entrées et colonnes, les unités
- Définir les règles attendues pour chaque colonne
  - Ex: âge doit être >0
- Définir l'utilisation finale des données
  - Contexte dans la diapo suivante

Contexte du dataset

- Crash d'avions
- Ce dataset doit être nettoyé et rendu utilisable pour du machine learning et du reporting
- Tri facile par pays et par modèle d'avion
- Il sera utilisé conjointement à d'autres datasets
- Foreign key sur les modèles d'avion
- Aucune entrée sur les modèles ne doit être nulle

OpenRefine

- OpenRefine va nous permettre d'identifier les traitements à effectuer
- Les traitement seront faire dans une second étape, avec pandas
- OpenRefine peut aussi faire des traitements, mais il est plus limité et moins souple
- Il possède un langage propre : GREL (semblable à excel)
- Préférer une autre solution pour les traitements
  - Python avec Pandas

Visualiser le dataset

- Ouvrez-le dans OpenRefine
- OpenRefine fonctionne par "facettes" (filtres)
- Une facette "Timeline" est spécialisé dans le traitement des dates
- Clic sur la colonne Date -> Facet -> Timeline facet
- On voit en bas "Time : 0", "Non time : 4825"
- La facette "timeline" ne reconnaît pas ces données, car elles sont en texte
- Edit cells -> Common transform -> To date
- On voit l'histogramme et on peut filtrer les données
- Une est encore en non Time, laquelle et pourquoi ?
- Mettez-la en favorite ("Star")

OpenRefine

- Créez les facettes pour Aboard et Fatalities. Que remarque-t-on ?
- Afficher uniquement les entrées avec données manquantes
- Faites une Text facet sur la colonne "Location". Combien de valeurs différentes ?
- On nous propose une "facet by choice counts", essayez
- L'axe X est le nombre d'occurrences, L'axe Y est la nombre de choix avec ce nombre d'occurrences.
- Quelles sont les 3 lieux les plus représentés ? Avec combien d'occurrence ?
- (blank):20, Sao Paulo:15, Moscou:15

OpenRefine

- Existe-t-il une entrée sans Location et sans Abord ni Fatalities ?
- Une seule
- Quelle genre de distribution ont l'air de suivre ces 3 colonnes ?
- Quel est le type d'avion le plus représenté ? Avec quel opérateur ?
- Quel crash de cet avion a fait le plus de victime ? Avec quel opérateur ?
  - Recherchez ce crash sur le web, pareil avec celui à 38 victimes

Clustering

- Faîtes un filtre de texte sur Type et tapez "boeing B"
- On observe "boeing B17G" et "boeing B-17G" et autres...
- Clustering = regrouper les termes similaires
- Edit cells -> Cluster and Edit
- Trouve les éléments très proches et propose de les remplacer
- Tout remplacer
- Regarder si d'autres méthodes trouvent encore des cluster
- Pareil avec Location et Operator

OpenRefine

- Quelles colonnes doivent être uniques ?
- Chaque entrée de la colonne index est-elle unique ?
- Index->Text Facet-> By count
- Les entrées en double sont-elles les mêmes ?
- Y a-t-il des index incorrects ?
- Index-> Numeric facet
- Y a-t-il des Registration en double ?
- Trop de valeurs pour afficher le résultat

Custom facets

- Facette customisée pour n'afficher que les doublons de Registration
- Langage GREL
- Allez sur la doc des fonctions GREL sur le site d'OpenRefine
- Regardez la doc de "facetCount" puis et créez cette facette
- facetCount(value, "value", "Registration") > 1
- Créer une colonne à partir de Registration (Add column based on that one)
- Mettre la même expression

OpenRefine

- Comment pourrait-on vérifier si une entrée avec même date, lieux et opérateur existe ?
- Créer une colonne supplémentaire synthétisant les trois autres
- value + " | " + cells["Date"].value + " | " + cells["Operator"].value
- Facette de texte sur cette colonne et chercher les doublons.
- Mettez en Star ceux qui sont incorrects
- All -> Facet by star
- Edit rows -> Remove matching rows
- Export -> CSV

DataFrame Pandas

- Objet permettant de manipuler des datasets
- Beaucoup de méthodes built-in
- Chargez et affichez les infos du CSV
  - read_csv
  - head(), info(), describe()
  - Sous quel format est lu la colonne "Date" ?

ydata-profiling

- Autre outil de profiling que OpenRefine
- Fonctionne sur pandas
- Regardez le prototype python de :

```
from ydata_profiling import ProfileReport
```

- Générez un rapport de profilage en html et ouvrez-le dans un navigateur
- Représentation visuelle simple et rapide du dataset, mais peu d'exploration possible dans l'ui
- Il faut manipuler le dataset en python et générer plusieurs rapports

Conclusion profiling

- Dataset clusterisé avec entrées en doublon supprimées
- Identification du nettoyage à faire:
- Enlever les entrées qui ont un Type à blank
- La colonne "Location" : séparer pays et villes
- Vérifier l'uniformisation des villes et pays
  - Par ex : USSR, Russia, russia, ru, ou Great-britain, Great britain, UK
  - Tout mettre au même format
- Pareil pour les noms des operators
- Séparer Type en modèle et version

Nettoyage

Pandas

- Essayez de convertir les dates en format de date
  - `pd.to_datetime(dataset['Date'])`
- Qu'observe-t-on ?
  - Un format de date est incorrect
- Décider du format : ici MM/DD/YYYY

```
pd.to_datetime(dataset['Date'], format='%m/%d/%Y', errors='coerce')
```

  - `coerce` : Force les erreurs en Not A Time (NaT)

Gestion des dates

- Trouvez et affichez les NaT
- Pourquoi sont-ils des NaT ?
- On a deux questions à se poser :
  - L'erreur avec la lettre va-t-elle se reproduire ?
  - Y aura-t-il d'autres formats de date ?
- Mettre en place un traitement automatique de ces deux problèmes pour toutes les entrées
- Lever une alerte s'il reste des NaT
- Que faire si on a à la fois MM-DD-YYYY et DD-MM-YYYY ?
  - Politique de gouvernance, alerte si mois > 12

Gestion des Types

- Drop toutes les entrée avec Type à null
- On veut pouvoir trouver facilement toutes les version d'un modèle
- Comment faire ?
- Séparer le type en modèle et version
  - Premier mot : modèle
  - Si un seul mot : version à null
  - Regardez la méthode .str.split
- Affichez les colonnes pour vérifier
- Mettre en lowercase pour éviter les doublons

Location

- Drop la colonne Index
- Convertir Location en str
- On veut accéder rapidement à tous les accidents ayant eu lieux dans un pays. Comment faire ?
- Mettre la colonne Location en lowerase (éviter les doublons Russia, russia)
- Séparer ville, pays, afficher ces colonnes
- Plusieurs formats différentes :
  - atlanticity new jersey
  - victoria british columbia, canada

Location

- Solution : couper la colonne en trois ou supprimer le county
- On va supprimer le county
- Ensuite, chercher les doublons communs :
  - Regrouper UK, United Kingdom, England, Great Britain
  - Car les crash en England sont aussi dans le UK : tout mettre dans le UK
- Que faire pour les counties américains ?
  - Les laisser tel quel : chaque county sera vu comme un pays

Gestion des valeurs manquantes

- Il manque souvent des valeurs numériques
- Aboard, Fatalities, Ground
- Que faire ?
- Deux choix :
  - Laisser manquantes si pas besoin de traitement
  - Supprimer les entrées
  - Remplacer les valeurs
- On peut remplacer de deux manières :
  - Valeur fixe : tout à zéro, la moyenne, la médiane, chercher valeur exacte …
  - Interpolation (linéaire, splines, …)

Gestion des valeurs manquantes

- Si on met les valeurs manquantes à une valeur fixe :
  - Changement de la moyenne et de la valeur médiane
- Si on les met à la moyenne : médiane variable, moyenne constante
- Si on les met à la médiane : moyenne variable, médiane constante
- Dans tous les cas la distribution va changer
- Interpoler permet de ne pas modifier la distribution
- Impossible ici : données non continues
- Par exemple : températures manquantes sur quelques heures : on peut interpoler pour calculer des valeurs approchées

Remplissage conditionnel

- On remplace les données manquantes par la moyenne de la colonne, arrondie à l'unité
- Problème : on veut une moyenne pour chaque modèle
- Moyenne de passagers et de mort **par modèle**, pas globale
- "Remplissage conditionnel par groupe"
- Regardez le prototype : dataset.groupby("Modele")["Aboard"].transform()
- Effectuez la transformation, vérifiez qu'il n'y a plus de valeur null
- Il en reste une, trouvez laquelle et pourquoi
- Il n'y a qu'un seul rochrbach dans tout le dataset : Moyenne NaN

Gestion des valeurs manquantes

- Allez chercher la vraie valeur et mettez manuellement à jour le dataset
- Exportez le dataset nettoyé en csv
- Chargez-le dans OpenRefine
- Regardez le détail des colonnes modèle / version et ville / pays
- Le modèle "De haviland" est scindé en deux
- Remplacer tous les "De haviland" en "de-haviland" en dur

Indicateurs KPI

- Key Performance Indicator
- Permet d'évaluer l'efficacité d'un processus par rapport à des objectifs définis
- Utilisé partout, pas qu'en informatique
- Par exemple : au moins 99,5 % d'uptime d'un service
- Lié à un objectif précis
- Mesurable, atteignable et pertinent
  - Vouloir 100% de uptime n'est pas atteignable.

Pipelines de nettoyage

- Script brut -> Création de pipelines
- "Mettre au propre"
- Pandas propose des outils pour la création de pipelines
- Enchaîne les opérations de traitement de donnes de manière propre et lisible
- Enchaîne les fonctions, built-in pandas ou custom
- Faire des fonctions pour le traitement de chaque fonction, sans sortie console mais avec alertes et commentaires

Indicateurs KPI

- Identifier les exigences de qualité puis définir les KPI
- Comment définir la complétude ?
- Quelles données doivent être uniques ?
- Quel est le temps de conservation de chaque donnée ?
  - Données météo ou de traffic : à quelle fréquence doit-on les archiver ?
  - Données stables : codes postaux, numéro de sécurité social, …
- Quels doivent être les types et intervalles de chaque donnée ?
- Quel format unique pour chaque donnée ?

Exemples de KPI

- 100% des données respectent le RGPD
- 0% des fichiers sont corrompus
- 100 % des dates sont au format DD/MM/YYYY
- Moins de 300ms de délai moyen pour accéder aux fichiers
- Toutes les données de plus de 5 ans doivent être archivées
- Moins de 80% du stockage occupé pour éviter la saturation
- Au moins 99,5% d'uptime pour l'API

L'objectif est de faire tourner des scripts vérifiant ça (devops)

Fin du nettoyage

- Décidez de la fin du nettoyage et explicitez les KPI
- Quels axes d'amélioration pour une deuxième version du traitement ?
- Quelles politiques de gouvernance des données peut-on envisager ?

Exemples de pipelines

```
  resultat = (df
      .dropna()
      .query('prix > 100')
      .assign(total=lambda x: x['prix'] * x['quantite'])
      .sort_values('total')
  )
```

```
resultat = (df
    .pipe(nettoyer)
    .pipe(ajouter_total)
    .pipe(filtrer, colonne="total", seuil=50)
    .sort_values("total", ascending=False)
)
```

Soda core

Soda-core

- Framework open source de testing de qualité des données
- Définir des règles de qualité des données (KPI)
- Exécuter des scans et lever des alertes
- Facilement intégrable à une pipeline intégration continue
- Peut se connecter à une base de données, charger un fichier, ...
- Critères de qualité définis dans un YAML
- Peut se lancer en console ou **en python avec des dataframe pandas**
- Souvent lancé à partir d'un script python

YAML Soda

- Deux fichiers
- **configuration.yml** : connexion aux datadabases et autres configs
- **checks.yml :** tests de qualité
- Structure du checks.yml par blocs de tests
  - "Check for Modele"
    - Tests sur la colonne Modele
  - "Check for Date"
    - Tests sur la colonne Date

Soda avec Pandas

- Soda core propose des foncitonnalités pour charger des dataframe pandas
- Mais soda n'est pas prévu ni optimisé pour ça !
- (J'ai galéré à le faire foncitonner)
- Il fonctionne mieux sur des databases
- Installez soda-core-duckdb
- duckdb permet d'instancier des petites bdd volatiles
- Partez du code fournit pour coder les checks de données manquantes et aberrantes

Orchestrateur (Dagster)

Orchestrateur

- Même principe que Jenkins pour l'IC
- Création de pipelines automatisées
- Pipeline simple :
  - Nettoyage
  - Checks
  - Alertes
- Apache Airflow, Dagster : Orchestrateur Open Source
- Se code en python

Dagster

- Outil très complet pour des pipelines data
- Code des "objets Dagster" en python
  - Beaucoup de variété
- Lance un UI avec ces objets
- Prendre le code fournit (provient d'un tuto youtube)
  - https://www.youtube.com/watch?v=sKqDq4TFbmY
- Lancez dagster dev –m orchestrator, allez dans assets
- Vous voyez vos deux assets, affichez le lineage
- Materialize all : lance la pipeline

Dagster

- Allez sur "Runs"
- Le run que vous venez de faire avec "Materialize all" est visible
- Cliquez pour afficher les infos du run
- Fermez le serveur
- Remplacez les deux fonctions par l'appel à vos scripts
- Importez les csv dans le code Dagster et passez-le en entrée de ces fonctions
- Lancez votre run

Donnée VS information

- Donnée = élément brut, sans interprétation
  - Type d'avion, nombre de passagers, ...
- Information = donnée traitée, structurée, mise en contexte
  - Répond à une question, donne un sens, permet une décision
- Donnée -> Information par le **traitement**
- L'information donne des connaissances et permet de **décisions**
- Exemple pour notre dataset : Les entrées sont les données bruts
- Des informations seraient :
  - "L'Europe représente 47,2% des accidents entre 1990 et 2010"
  - "Le taux de mortalité moyen des accidents d'Airbus est de 67%"
  - "Le premier crash Boeing enregistré est en 1928"

Niveau de granularité

- Plusieurs niveaux de qualité des données d'une entreprise
- Premier niveau : "Niveau stratégique"
  - Vue d'ensemble globale
  - KPI agrégés
  - Rapports trimestriels / annuels
- Second niveau : "Niveau tactique"
  - Qualité par processus ou source de données
  - Rapports mensuels
- Troisième niveau : "Niveau opérationnel"
  - Temps réel
  - Enregistrement par enregistrement, utilisation par utilisation

Informations

- Quels informations peut-on tirer de notre dataset ?
- Nombre moyen de décès par modèle ou fabricant
- Pourcentage de décès par nombre total de passager pour chaque modèle / fabricant
- Pareil pour les operators, les localités
- Prévalence d'un opérateur pour chaque modèle
- Lien entre corrélation et causation !
- https://www.kaggle.com/code/vedumrajkar/airplane-crashes-analysis

Réconciliation des données

- Comparaison et harmonisation des données venant de plusieurs sources
- Étape 1 : Identification des sources de données à réconcilier
- Étape 2 : Définition des règles de réconciliation
  - Quelle source fait foi ?
  - Clés de jointure
- Les clés de jointures sont les foreign keys permettant de relier deux tables
- Étape 3 : Comparaison et analyse des écarts
- Étape 4 : Résolution et documentation

Réconciliation via online datas

- Dans une colonne OpenRefine: cliquer sur reconciliation
- Propose WikiData
- Browse WikiData automatiquement et cherche le "best candidate"
- Propose plusieurs sources de données au sein de WikiData
- Essayez avec quelques colonnes (notamment modèle et version)
- Possibilité de rajouter d'autres sources de données que OpenRefine
- Permet uniformisation des formats par ceux de wikidata

Réconciliation des Type

- Dans la dataset original : reconcile avec wikidata
- Très chronophage, le faire avec un petit nombre d'entrées
- Puis, scripts de nettoyage et prétraitement
- Modèles et versions harmonisés

Confidentialité des données

- Données publiques ou privées ?
- Comment gérer les droits d'accès ?
- Association d'un ID au nom des clients, l'ID est rendu public, table de conversion ID/nom-prénom privée
- Stockage des données uniquement nécessaires
- Chiffrement
- Retrait d'application (droit à l'oubli) :
  - Le RGPD donne le droit à toute personne de demander à toute personne le droit de supprimer ses données personnelles

Data management

- Définir les enjeux et besoin des données
- Définir les KPI et les différents niveaux de qualité des données
- Lien avec les processus métiers
- Règles de confidentialité et d'archivage
- Politiques de réconciliation des données
- Stockage et sécurité