

MISE EN PLACE D'UN DATA LAKE POUR L'ANALYSE DU MARCHÉ DU TRAVAIL

BOUAZIZI M., CASANOVA S., CONDE K., RAMOS Y.

184

Nb entreprises

254

Nb offres

79

Nb de villes

4115

Nb candidates

Nb offres de travail



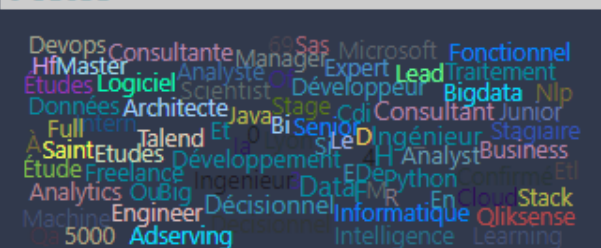
Sentiment de commentaire moyen



Entreprises



Postes



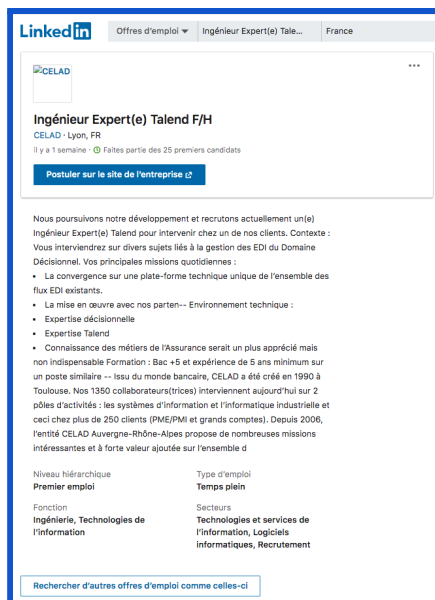
MISE EN PLACE D'UN DATA LAKE POUR L'ANALYSE DU MARCHÉ DU TRAVAIL

1. CONTEXTE

Ce projet consiste en la mise en place d'un data lake de données liées au marché de travail, plus spécifiquement, les *avis sur les entreprises*, les *offres de travail* et les *entreprises*.

La source de données est composé pour fichiers html distribués de cette manière :

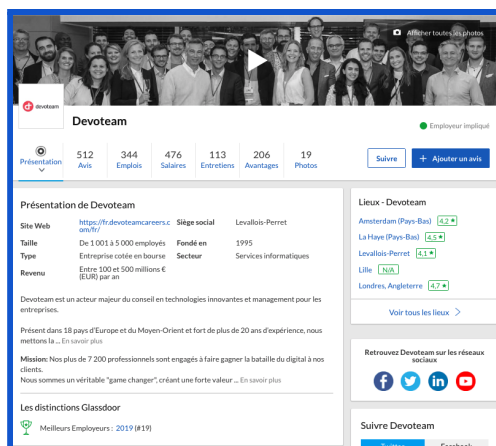
- **Offres de travail** : 288 fichiers htm



Informations :

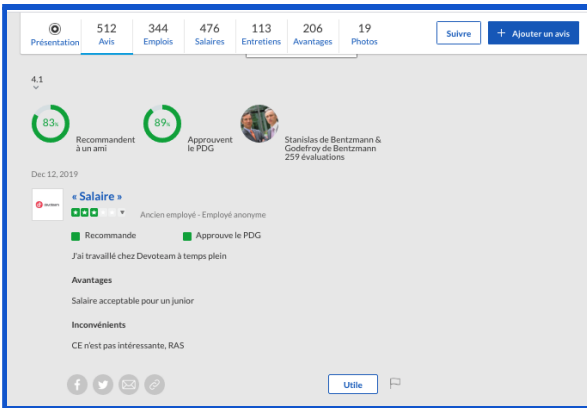
- Poste
- Entreprise
- Lieu
- Date de publication
- Description de poste
- Hiérarchie
- Fonctions
- Type d'emploi
- Secteurs

- **Entreprise**: 173 fichiers html



Informations :

- Entreprise
- Page web
- Taille
- Type
- Siege
- Fonctions
- Revenus
- Secteurs
- Avis
- **Avis sur les entreprises**: 242 fichiers html



Informations :

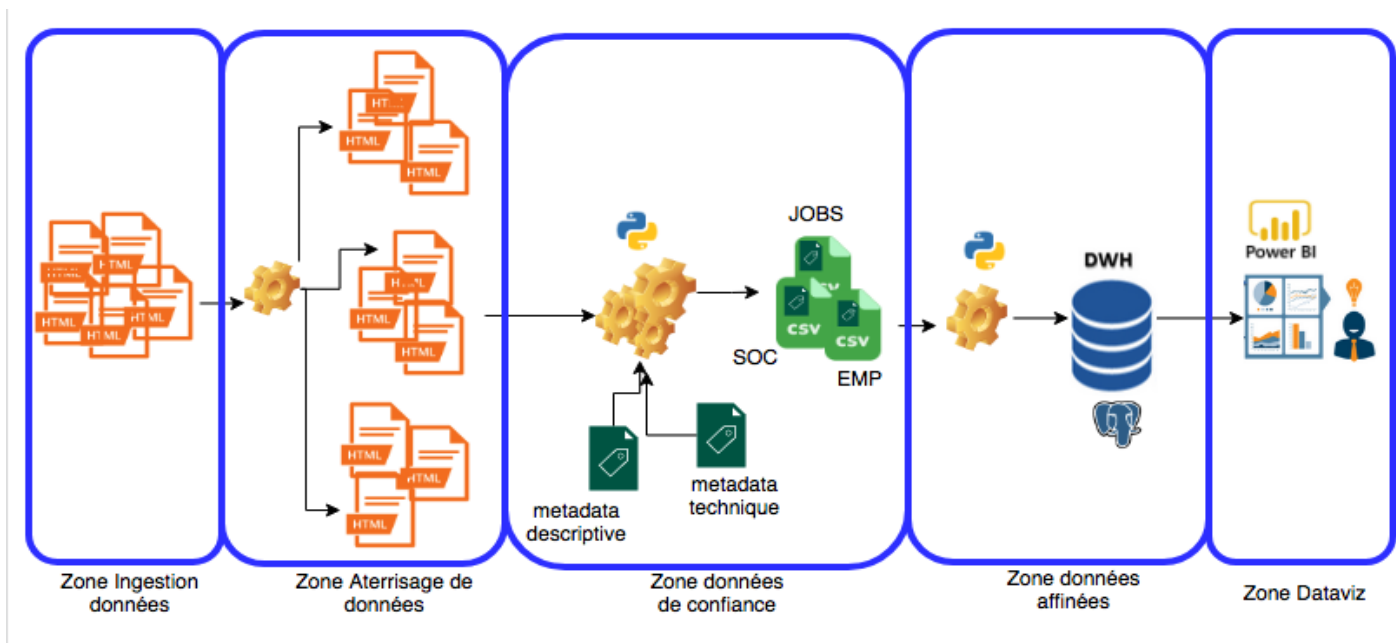
- Entreprise
- Commentaire
- Avantages
- Inconvénients
- Lieu
- Date d'avis
- Recommend l'entreprise
- Status employé

La mise en place du data lake comprend les étapes :

- **Ingestion des données :** Récupération des données
- **Préparation de données :** Métadonnées descriptives et techniques
- **Raffinage des données :** Archivage, stockage et modélisation datawarehouse
- **Dataviz**

2. ARCHITECTURE PROPOSÉE

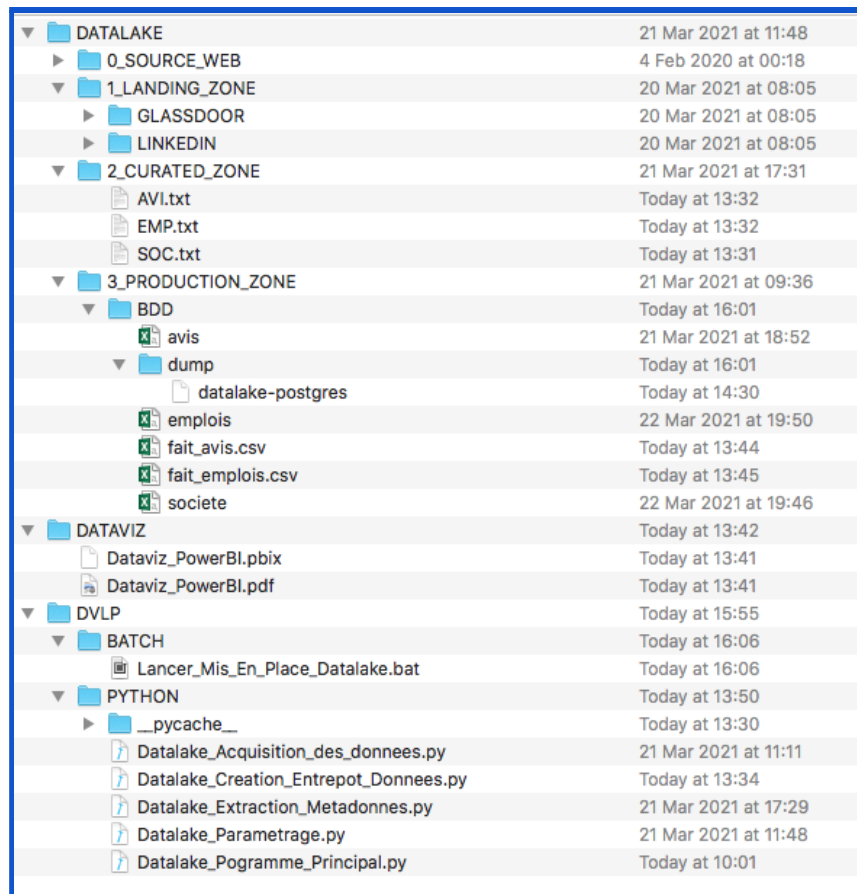
L'architecture proposé pour le projet est la suivante :



Les outils utilisés sont :

- Langage pour le codage : **PYTHON**
- Fichiers de données : **CSV**
- Langage de Script : **Batch**
- Base de données : **PostgreSQL**
- Langage de requête : **SQL**
- Data Visualization : **Power BI**

La structure du projet est :



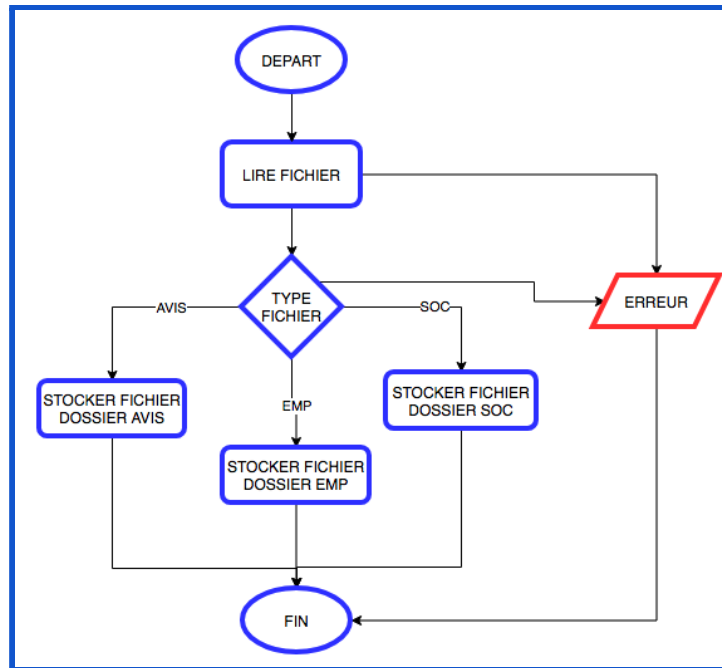
DATA LAKE	21 Mar 2021 at 11:48
0_SOURCE_WEB	4 Feb 2020 at 00:18
1_LANDING_ZONE	20 Mar 2021 at 08:05
GLASSDOOR	20 Mar 2021 at 08:05
LINKEDIN	20 Mar 2021 at 08:05
2_CURATED_ZONE	21 Mar 2021 at 17:31
AVI.txt	Today at 13:32
EMP.txt	Today at 13:32
SOC.txt	Today at 13:31
3_PRODUCTION_ZONE	21 Mar 2021 at 09:36
BDD	Today at 16:01
avis	21 Mar 2021 at 18:52
dump	Today at 16:01
datalake-postgres	Today at 14:30
emplois	22 Mar 2021 at 19:50
fait_avis.csv	Today at 13:44
fait_emplois.csv	Today at 13:45
societe	22 Mar 2021 at 19:46
DATAVIZ	Today at 13:42
Dataviz_PowerBI.pbix	Today at 13:41
Dataviz_PowerBI.pdf	Today at 13:41
DVLP	Today at 15:55
BATCH	Today at 16:06
Lancer_Mis_En_Place_Datalake.bat	Today at 16:06
PYTHON	Today at 13:50
__pycache__	Today at 13:30
Datalake_Acquisition_des_donnees.py	21 Mar 2021 at 11:11
Datalake_Creation_Entrepot_Donnees.py	Today at 13:34
Datalake_Extraction_Metadonnees.py	21 Mar 2021 at 17:29
Datalake_Parametrage.py	21 Mar 2021 at 11:48
Datalake_Pogramme_Principal.py	Today at 10:01

3. MISE EN PLACE DE DATA LAKE

3.1 INGESTION DES DONNÉES

La première tâche à faire est la récupération de sources. Dans notre cas, nous avons écrit un code qui parcourt chaque fichier et le stocke dans un dossier lié à son thématique, par exemple, les fichiers avec la particule %AVIS% sont liés avec la thématique de **AVIS**, lesquels avec %SOC% avec **SOCIÉTÉ**, et finalement les fichiers avec %EMP% sont liés avec **EMPLOIS**.

Sur la figure suivante, nous trouvons le workflow du code :



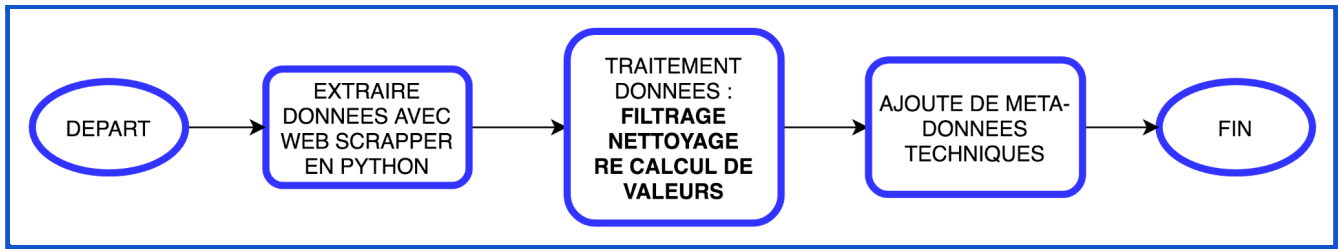
Ensuite le code fait en python :

```

import os, fnmatch
import shutil
from Datalake_Parametrage import myPathRoot_DATASOURCE
from Datalake_Parametrage import myPathRoot_LANDINGZONE
def Recuperation_Fichiers_HTML_SOURCE(ChoixDebug=True, TypeDeFichier=None, OrigineDuFichier=None):
    if TypeDeFichier == 'EMP':
        mySousRepertoire = 'EMP'
        myFiltre = 'INFO-EMP'
    elif TypeDeFichier == 'SOC':
        mySousRepertoire = 'SOC'
        myFiltre = 'INFO-SOC'
    elif TypeDeFichier == 'AVI':
        mySousRepertoire = 'AVI'
        myFiltre = 'AVIS-SOC'
    else:
        pass
    myPattern = "*" + myFiltre + "*.html" #-- Pour le parametre 'AVI' par exemple cela generera : myPattern = "*" + 'AVI-SOC' + "*.html"
    if (ChoixDebug):
        myListOfFile = []
        for myFileName in myListOfFileALL:
            if fnmatch.fnmatch(myFileName, myPattern):
                myListOfFile.append(myFileName)
        myPathHtmlLANDINGZONE = myPathRoot_LANDINGZONE + "/" + OrigineDuFichier + "/" + mySousRepertoire
        for myFileNameTmp in myListOfFile:
            shutil.copy(myPathHtmlDATASOURCE + "/" + myFileNameTmp, myPathHtmlLANDINGZONE + "/" + myFileNameTmp)
    return(True)
  
```

3.2 PRÉPARATION DE DONNÉES

Dans la partie de préparation de données, nous avons fait 3 actions :



La première est l'écriture du **web scraper** qui permet de récupérer l'information de chaque page selon le besoin. Le web scraper a été fait en python en utilisant la librairie **BeatifulSoup**.

Les données récupérées pour la thématique **d'emploi** sont :

Poste, entreprise, location, date_publication, nro_candidates, description_job, hierarchie, type_emploi, fonctions et secteurs.

Code utilisé pour la récupération de la donné du poste :

```

def Get_libelle_emploi_EMP(Soup):
    myTest = Soup.find_all('h1', attrs = {'class':'topcard__title'})
    if (myTest == []):
        Result = "NULL"
    else:
        myTest = str(myTest[0].text)
        if (myTest == []):
            Result = "NULL"
        else:
            Result = myTest.replace(";", ";").replace('\n', ' ').replace('\r', '')
    return(Result)
  
```

Pour les **avis**, nous avons récupéré :

Entreprise, date, review_titre, status_employé, lieu, recommandé, commentaire, avantage, inconvenient

Code pour la récupération des inconvénients :

```

def Get_inconvénients(soup2):
    myTest2 = soup2.find_all('div', attrs = {'class':'mt-md common__EiReviewTextStyles__allowLineBreaks'})
    if (myTest2 == []):
        return "Pas d'avantages pour les employés"
    else:
        leng = len(myTest2)
        if (leng == 2):
            Result = myTest2[1].contents[1].text.replace(";", ";").replace('\n', ' ').replace('\r', '')
        else:
            Result = "Pas d'avantages pour les employés"
    return (Result)
  
```

Et pour les sociétés :

Nom_entreprise, nro_avis, site_web, taille, date_fondation, secteur, revenu, type_entreprise

Code pour la récupération des nro_avis :

```
mySoup.find_all('a', attrs = {'data-label':"Avis"})[0].span.contents[0]
```

La deuxième activité est de traiter les données, ça veut dire : **filtrage, nettoyage et de re calculs de valeurs**. Exemple de ces traitements est le **calcul de sentiment pour les commentaires et les review_titre des avis** avec la librairie **TextBlob** :

```
review_titre=TextBlob(line[6], pos_tagger = PatternTagger(), analyzer= PatternAnalyzer()).sentiment[0]
```

Le calcul de mot plus fréquent dans la description_job avec la librairie **Counter**:

```
mostW = Counter(word for word in filter(str.isalpha,line[9].lower().split()) if word not in stops).most_common(1)
description_job= mostW.pop(0)[0]
```

Des autres exemples sont le parsing de type de données à string avec **str** et le remplacement des caractères de saute de ligne ou autres qui génèrent de bruit dans la donnée.

La troisième correspond à intégrer les métadonnées techniques tels que : **date d'ingestion, localisation du fichier, niveau d'accessibilité (entre 0 et 3, avec 0 comment public)**. Le code pour générer la date d'ingestion est :

```
from datetime import datetime
def Get_datetime_ingestion_AVI():
    Result = str(datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
    return(Result)
```

A continuation, nous présentons des extraits de code pour la récupération de données pour chaque thématique et aussi des extraits de résultats (fichiers csv). Le texte souligné en bleu correspond aux metadonnées techniques et le texte en jaune sont les données récupérés.

EMP (emplois) :

```
def Generation_Fichiers_avec_Metadonnees_EMP():
    myFilePathName = myPathCuratedZone + "EMP.txt"
    myFilePtr = open(myFilePathName, "w", encoding = "utf-8")
    myListeDeLigneAEcrire = []

    for myEntry in myListOfFileEMP :
        f = open(myPathHtmlEMP+myEntry, "r", encoding="utf8")
        myHTMLContents = f.read()
        f.close()
        mySoup = BeautifulSoup(myHTMLContents, 'xml')
        cle_unique=str(random.getrandbits(128))
        emplacement_source=myEntry
        datetime_ingestion=str(Get_datetime_ingestion_AVI())
        privacy_level="0"
```

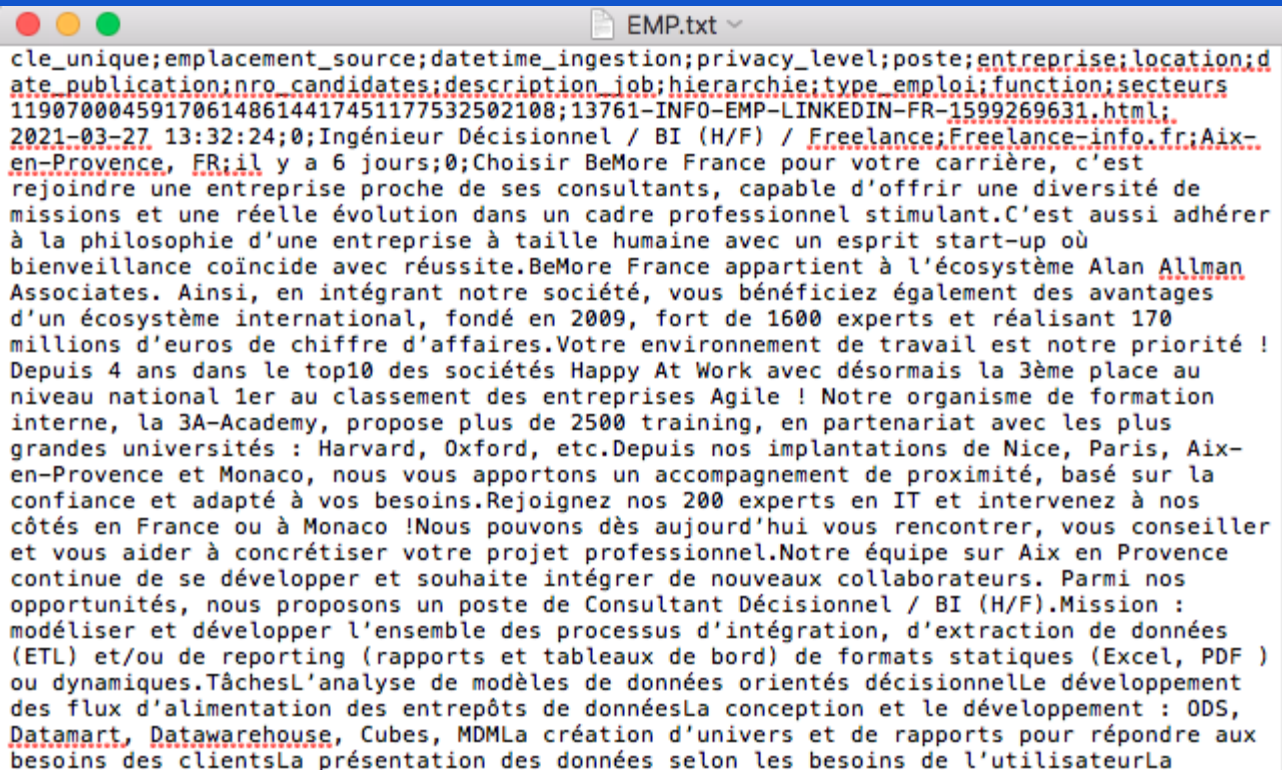


```

poste = Get_libelle_emploi_EMP(mySoup)
entreprise = Get_nom_entreprise_EMP(mySoup)
location = Get_ville_emploi_EMP(mySoup)
date_publication = Get_date_emploi_EMP(mySoup)
nro_candidates = Get_candidats_emploi_EMP(mySoup)
description_job = mySoup.find_all('div', attrs = {'class': 'description__text description__text--rich'})[0].text
if len(mySoup.find_all('ul', attrs = {'class': 'job-criteria__list'})) [0].contents) == 4 :
    hierarchie = str(mySoup.find_all('ul', attrs = {'class': 'job-criteria__list'})[0].contents[0].span.contents[0])
    type_emploi = str(mySoup.find_all('ul', attrs = {'class': 'job-criteria__list'})[0].contents[1].span.contents[0])
    fonction = str(mySoup.find_all('ul', attrs = {'class': 'job-criteria__list'})[0].contents[2].span.contents[0])
    secteurs = str(mySoup.find_all('ul', attrs = {'class': 'job-criteria__list'})[0].contents[3].span.contents[0])
myListeDeLigneAEcrire.append(values "\n")

myFilePtr.writelines(myListeDeLigneAEcrire)
myFilePtr.close()
return (True)

```



```

cle_unique;emplacement_source;datetime_ingestion;privacy_level;poste;entreprise;location;date_publication;nro_candidates;description_job;hierarchie;type_emploi;fonction;secteurs
119070004591706148614417451177532502108;13761-INFO-EMP-LINKEDIN-FR-1599269631.html;
2021-03-27 13:32:24;0;Ingénieur Décisionnel / BI (H/F) / Freelance;Freelance-info.fr;Aix-en-Provence, FR;il y a 6 jours;0;Choisir BeMore France pour votre carrière, c'est rejoindre une entreprise proche de ses consultants, capable d'offrir une diversité de missions et une réelle évolution dans un cadre professionnel stimulant.C'est aussi adhérer à la philosophie d'une entreprise à taille humaine avec un esprit start-up où bienveillance coïncide avec réussite.BeMore France appartient à l'écosystème Alan Allman Associates. Ainsi, en intégrant notre société, vous bénéficiez également des avantages d'un écosystème international, fondé en 2009, fort de 1600 experts et réalisant 170 millions d'euros de chiffre d'affaires.Votre environnement de travail est notre priorité ! Depuis 4 ans dans le top10 des sociétés Happy At Work avec désormais la 3ème place au niveau national 1er au classement des entreprises Agile ! Notre organisme de formation interne, la 3A-Academy, propose plus de 2500 training, en partenariat avec les plus grandes universités : Harvard, Oxford, etc.Depuis nos implantations de Nice, Paris, Aix-en-Provence et Monaco, nous vous apportons un accompagnement de proximité, basé sur la confiance et adapté à vos besoins.Rejoignez nos 200 experts en IT et intervenez à nos côtés en France ou à Monaco !Nous pouvons dès aujourd'hui vous rencontrer, vous conseiller et vous aider à concrétiser votre projet professionnel.Notre équipe sur Aix en Provence continue de se développer et souhaite intégrer de nouveaux collaborateurs. Parmi nos opportunités, nous proposons un poste de Consultant Décisionnel / BI (H/F).Mission : modéliser et développer l'ensemble des processus d'intégration, d'extraction de données (ETL) et/ou de reporting (rapports et tableaux de bord) de formats statiques (Excel, PDF ) ou dynamiques.TâchesL'analyse de modèles de données orientés décisionnelle développement des flux d'alimentation des entrepôts de donnéesLa conception et le développement : ODS, Datamart, Datawarehouse, Cubes, MDMLa création d'univers et de rapports pour répondre aux besoins des clientsLa présentation des données selon les besoins de l'utilisateurLa

```

AVI (avis des entreprises) :

```

def Generation_Fichiers_avec_Metadonnees_AVI()
    myFilePathName = myPathCuratedZone + "AVI.txt"
    myFilePtr = open(myFilePathName, "w", encoding = "utf-8")
    myListDeLigneAEcrire = []

    for myEntry in myListOfFileAVI :
        f = open(myPathHtmlAVI+myEntry, "r", encoding="utf8")
        myHTMLContents = f.read()
        f.close()
        mySoup = BeautifulSoup(myHTMLContents, 'lxml')
        avis = mySoup.find_all('li', attrs = {'class': 'empReview'})

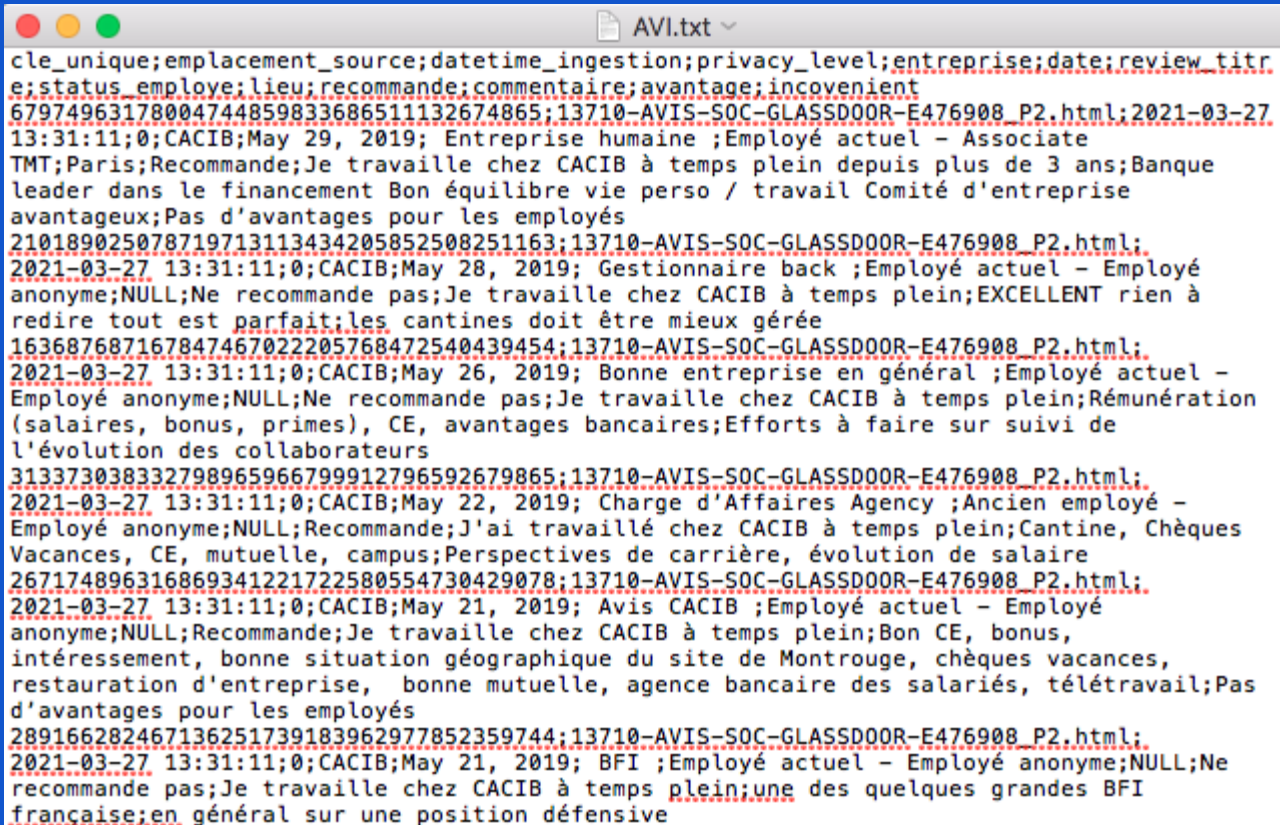
```



```

myListTab=[]
if (avis == []) :
    print("NULL")
else:
    for x in range(0, len(avis)) :
        if x == 0:
            myListTab[0] = ["0"]
        else:
            soup2 = BeautifulSoup(str(avis[x]), 'xml')
            cle_unique=str(random.getrandbits(128))
            emplacement_source=myEntry
            datetime_ingestion=str(Get_datetime_ingestion_AVI())
            privacy_level="0"
            entreprise=Get_nom_entreprise_AVI(mySoup)
            date=Get_date(soup2)
            review_titre=Get_review_titre(soup2)[1:-1]
            status_employe=Get_employe_actuel(soup2)
            lieu=Get_ville_employe(soup2)
            recommande=Get_recommande(soup2)
            commentaire=Get_commentaire(soup2)
            avantage=Get_avantages(soup2)
            inconvenient=Get_inconvenients(soup2)
            myFilePtr.writelines(myListeDeLigneAEcrire)
            myFilePtr.close()
    return (True)

```



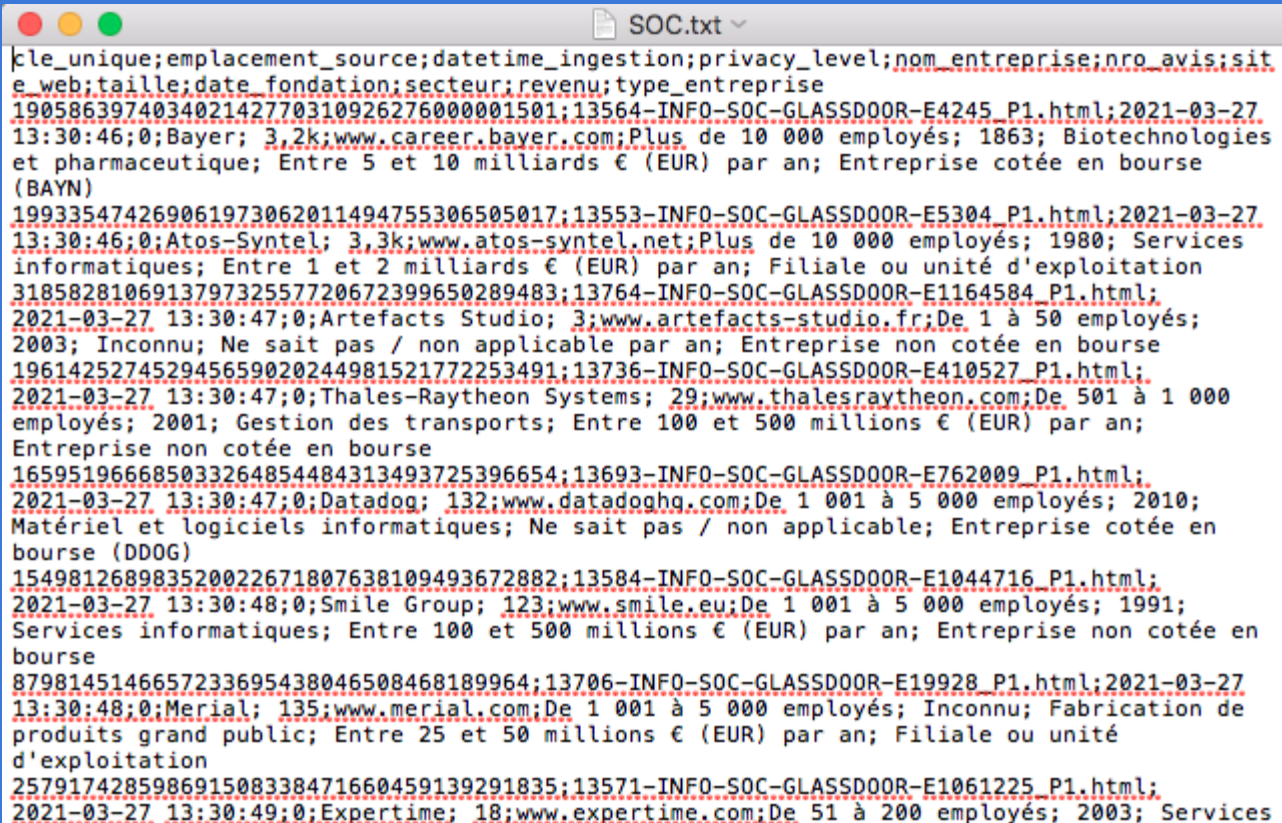
```

cle_unique;emplacement_source;datetime_ingestion;privacy_level;entreprise;date;review_titr
e;status_employe;lieu;recommande;commentaire;avantage;inconvenient
67974963178004744859833686511132674865;13710-AVIS-SOC-GLASSD00R-E476908_P2.html;2021-03-27
13:31:11;0;CACIB;May 29, 2019; Entreprise humaine ;Employé actuel - Associate
TMT;Paris;Recommande;Je travaille chez CACIB à temps plein depuis plus de 3 ans;Banque
leader dans le financement Bon équilibre vie perso / travail Comité d'entreprise
avantageux;Pas d'avantages pour les employés
210189025078719713113434205852508251163;13710-AVIS-SOC-GLASSD00R-E476908_P2.html;
2021-03-27 13:31:11;0;CACIB;May 28, 2019; Gestionnaire back ;Employé actuel - Employé
anonyme;NULL;Ne recommande pas;Je travaille chez CACIB à temps plein;EXCELLENT rien à
redire tout est parfait;les cantines doit être mieux gérée
163687687167847467022205768472540439454;13710-AVIS-SOC-GLASSD00R-E476908_P2.html;
2021-03-27 13:31:11;0;CACIB;May 26, 2019; Bonne entreprise en général ;Employé actuel -
Employé anonyme;NULL;Ne recommande pas;Je travaille chez CACIB à temps plein;Rémunération
(salaires, bonus, primes), CE, avantages bancaires;Efforts à faire sur suivi de
l'évolution des collaborateurs
313373038332798965966799912796592679865;13710-AVIS-SOC-GLASSD00R-E476908_P2.html;
2021-03-27 13:31:11;0;CACIB;May 22, 2019; Charge d'Affaires Agency ;Ancien employé -
Employé anonyme;NULL;Recommande;J'ai travaillé chez CACIB à temps plein;Cantine, Chèques
Vacances, CE, mutuelle, campus;Perspectives de carrière, évolution de salaire
267174896316869341221722580554730429078;13710-AVIS-SOC-GLASSD00R-E476908_P2.html;
2021-03-27 13:31:11;0;CACIB;May 21, 2019; Avis CACIB ;Employé actuel - Employé
anonyme;NULL;Recommande;Je travaille chez CACIB à temps plein;Bon CE, bonus,
intérêt, bonne situation géographique du site de Montrouge, chèques vacances,
restauration d'entreprise, bonne mutuelle, agence bancaire des salariés, télétravail;Pas
d'avantages pour les employés
289166282467136251739183962977852359744;13710-AVIS-SOC-GLASSD00R-E476908_P2.html;
2021-03-27 13:31:11;0;CACIB;May 21, 2019; BFI ;Employé actuel - Employé anonyme;NULL;Ne
recommande pas;Je travaille chez CACIB à temps plein;une des quelques grandes BFI
française;en général sur une position défensive

```

SOC (société) :

```
def Generation_Fichiers_avec_Metadonnees_SOC():
    myFilePathName = myPathCuratedZone + "SOC.txt"
    myFilePtr = open(myFilePathName, "w", encoding = "utf-8")
    myListeDeLigneAEcrire = []
    for myEntry in myListOfFileSOC :
        f = open(myPathHtmlSOC+myEntry, "r", encoding="utf8")
        myHTMLContents = f.read()
        f.close()
        mySoup = BeautifulSoup(myHTMLContents, 'lxml')
        if len(mySoup.find_all('div', attrs = {'class':"infoEntity"})) == 7 :
            cle_unique=str(random.getrandbits(128))
            emplacement_source=myEntry
            datetime_ingestion=str(Get_datetime_ingestion_AVI())
            privacy_level="0"
            nom_entreprise=(mySoup.find('h1')['data-company'])
            nro_avis=str(mySoup.find_all('a', attrs = {'data-label':"Avis"})[0].span.contents[0])
            site_web=rmySoup.find_all('div', attrs = {'class':"infoEntity"})[0].span.contents[0]).group(1)
            taille= str(mySoup.find_all('div', attrs = {'class':"infoEntity"})[2].span.contents[0])
            date_fondation=str(mySoup.find_all('div', attrs = {'class':"infoEntity"})[3].span.contents[0])
            secteur=str(mySoup.find_all('div', attrs = {'class':"infoEntity"})[5].span.contents[0])
            revenu=str(mySoup.find_all('div', attrs = {'class':"infoEntity"})[6].span.contents[0])
            type_entreprise=str(mySoup.find_all('div', attrs = {'class':"infoEntity"})[4].span.contents[0])
        myFilePtr.writelines(myListeDeLigneAEcrire)
    myFilePtr.close()
    return (True)
```



SOC.txt

cle_unique;emplacement_source;datetime_ingestion;privacy_level;nom_entreprise;nro_avis;site_web;taille;date_fondation;secteur;revenu;type_entreprise

190586397403402142770310926276000001501;13564-INFO-SOC-GLASSD00R-E4245_P1.html;2021-03-27 13:30:46;0;Bayer; 3,2k;www.career.bayer.com;Plus de 10 000 employés; 1863; Biotechnologies et pharmaceutique; Entre 5 et 10 milliards € (EUR) par an; Entreprise cotée en bourse (BAYN)

199335474269061973062011494755306505017;13553-INFO-SOC-GLASSD00R-E5304_P1.html;2021-03-27 13:30:46;0;Atos-Syntel; 3,3k;www.atos-syntel.net;Plus de 10 000 employés; 1980; Services informatiques; Entre 1 et 2 milliards € (EUR) par an; Filiale ou unité d'exploitation

318582810691379732557720672399650289483;13764-INFO-SOC-GLASSD00R-E1164584_P1.html; 2021-03-27 13:30:47;0;Artefacts Studio; 3;www.artefacts-studio.fr;De 1 à 50 employés; 2003; Inconnu; Ne sait pas / non applicable par an; Entreprise non cotée en bourse

196142527452945659020244981521772253491;13736-INFO-SOC-GLASSD00R-E410527_P1.html; 2021-03-27 13:30:47;0;Thales-Raytheon Systems; 29;www.thalesraytheon.com;De 501 à 1 000 employés; 2001; Gestion des transports; Entre 100 et 500 millions € (EUR) par an; Entreprise non cotée en bourse

165951966685033264854484313493725396654;13693-INFO-SOC-GLASSD00R-E762009_P1.html; 2021-03-27 13:30:47;0;Datadog; 132;www.datadoghq.com;De 1 001 à 5 000 employés; 2010; Matériel et logiciels informatiques; Ne sait pas / non applicable; Entreprise cotée en bourse (DDOG)

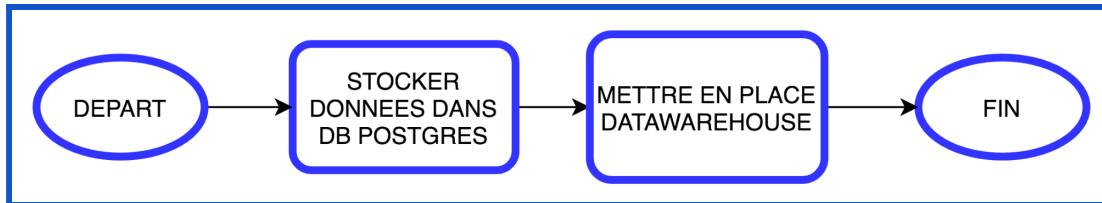
154981268983520022671807638109493672882;13584-INFO-SOC-GLASSD00R-E1044716_P1.html; 2021-03-27 13:30:48;0;Smile Group; 123;www.smile.eu;De 1 001 à 5 000 employés; 1991; Services informatiques; Entre 100 et 500 millions € (EUR) par an; Entreprise non cotée en bourse

8798145146657233695438046508468189964;13706-INFO-SOC-GLASSD00R-E19928_P1.html;2021-03-27 13:30:48;0;Merial; 135;www.merial.com;De 1 001 à 5 000 employés; Inconnu; Fabrication de produits grand public; Entre 25 et 50 millions € (EUR) par an; Filiale ou unité d'exploitation

257917428598691508338471660459139291835;13571-INFO-SOC-GLASSD00R-E1061225_P1.html; 2021-03-27 13:30:49;0;Expertime; 18;www.expertime.com;De 51 à 200 employés; 2003; Services

3.3 RAFFINAGE DES DONNÉES

Dans la partie de raffinement de données, nous avons fait 2 actions :



La première partie est le stockage de données récupérées et transformées. Nous avons utilisé la librairie **PSYCOP2** pour stocker les données dans un db de type **postgres**. Nous avons créé initialement 3 tables: SOCIÉTÉ, AVIS et EMPLOI :

Code utilisé pour créations des tables :

```
def Initialization_Database () :
    con = psycopg2.connect(database="BASE_CURATED_ZONE", user="postgres", password="admin", host="127.0.0.1", port="5433")
    cur = con.cursor()
    cur.execute("DROP TABLE IF EXISTS DIM_SOCIETE;");
    cur.execute("DROP TABLE IF EXISTS AVIS;");
    cur.execute("DROP TABLE IF EXISTS EMPLOI;");
    cur.execute("DROP TABLE IF EXISTS FAIT_EMPLOIS;");
    cur.execute("DROP TABLE IF EXISTS FAIT_AVIS;");

    cur.execute("""CREATE TABLE DIM_SOCIETE
        (CLE DECIMAL PRIMARY KEY NOT NULL,
        EMPLACEMENT_SOURCE CHAR(500),
        DATETIME_INGESTION DATE,
        PRIVACY_LEVEL CHAR(50),
        NOM_ENTREPRISE CHAR(500),
        NRO_AVIS CHAR(500),
        SITE_WEB CHAR(500),
        TAILLE CHAR(500),
        DATE_FONDATION CHAR(500),
        SECTEUR CHAR(500),
        REVENU CHAR(500),
        TYPE_ENTREPRISE CHAR(500)
        );""")
    print("Table SOCIETE created successfully")

    cur.execute("""CREATE TABLE EMPLOI
        (CLE DECIMAL PRIMARY KEY NOT NULL,
        EMPLACEMENT_SOURCE CHAR(500),
        DATETIME_INGESTION DATE,
        PRIVACY_LEVEL CHAR(50),
        POSTE CHAR(500),
        ENTREPRISE CHAR(500),
        LOCATION CHAR(500),
        DATE_PUBLICATION CHAR(50),
        NRO_CANDIDATES INT,
        DESCRIPTION_JOB CHAR(5000),
        HIERARCHIE CHAR(5000),
        TYPE_EMPLOI CHAR(5000),
        FUNCTION CHAR(5000),
        SECTEURS CHAR(5000)
        );""")
    print("Table EMPLOI created successfully")
```

```

cur.execute("""CREATE TABLE AVIS
(CLE DECIMAL PRIMARY KEY NOT NULL,
EMPLACEMENT_SOURCE CHAR(500),
DATETIME_INGESTION DATE,
PRIVACY_LEVEL CHAR(50),
ENTREPRISE CHAR(500),
DATE_AVIS DATE,
REVIEW_TITRE FLOAT,
STATUS_EMPLOYE CHAR(500),
LIEU CHAR(500),
RECOMMANDE CHAR(500),
COMMENTAIRE FLOAT,
AVANTAGE CHAR(5000),
INCONVENIENT CHAR(5000)
);""")
print("Table AVIS created successfully")
con.commit()
con.close()
return (True)

```

Postérieurement, nous avons stocké l'information dans chaque table :

Code utilisé pour stocker les données :

```

def Insert_Donnees_SOC() :
    con = psycopg2.connect(database="BASE_CURATED_ZONE", user="postgres", password="admin", host="127.0.0.1", port="5433")
    cur = con.cursor()
    myFilePathName = myPathRoot_CURRATEDZONE + "SOC.txt"
    myFilePtr = open(myFilePathName, "r", encoding="utf-8", errors="ignore")
    myFileContents = myFilePtr.readlines()
    del myFileContents[0]
    for myLineRead in myFileContents:
        line = myLineRead.split(";")
        cle_unique = int(line[0])
        emplacement_source=line[1]
        datetime_ingestion=line[2]
        privacy_level=line[3]
        nom_entreprise=line[4]
        nro_avis=line[5]
        site_web=line[6]
        taille=line[7]
        date_fondation=line[8]
        secteur=line[9]
        revenu=line[10]
        type_entreprise=line[11].replace("\n","")
        cur.execute("INSERT INTO DIM_SOCIETE VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(cle_unique,
emplacement_source, datetime_ingestion, privacy_level, nom_entreprise,nro_avis, site_web,taille, date_fondation,
secteur,revenu, type_entreprise))
    myFilePtr.close()
    con.commit()
    con.close()
    return (True)

```

```

def Insert_Donnees_EMP() :
    con = psycopg2.connect(database="BASE_CURATED_ZONE", user="postgres", password="admin", host="127.0.0.1", port="5433")
    cur = con.cursor()
    myFilePathName = myPathRoot_CURRATEDZONE + "EMP.txt"
    myFilePtr = open(myFilePathName, "r", encoding="utf-8", errors="ignore")
    myFileContents = myFilePtr.readlines()
    del myFileContents[0]

```

```

for myLineRead in myFileContents:
    line = myLineRead.split(";")
    cle_unique = int(line[0])
    emplacement_source=line[1]
    datetime_ingestion=line[2]
    privacy_level=line[3]
    poste=line[4]
    entreprise=line[5]
    location=line[6]
    date_publication=line[7]
    nro_candidates=line[8].split()[0]
    description_job= line[9]
    hierarchie=line[10]
    type_emploi=line[11]
    fonction=line[12]
    secteurs=line[13]
    cur.execute("INSERT INTO EMPLOI VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
        (cle_unique,emplacement_source,datetime_ingestion,privacy_level,poste,entreprise,location,date_
publication,nro_candidates,description_job,hierarchie,type_emploi,fonction,secteurs))
    myFilePtr.close()
con.commit()
con.close()
return (True)

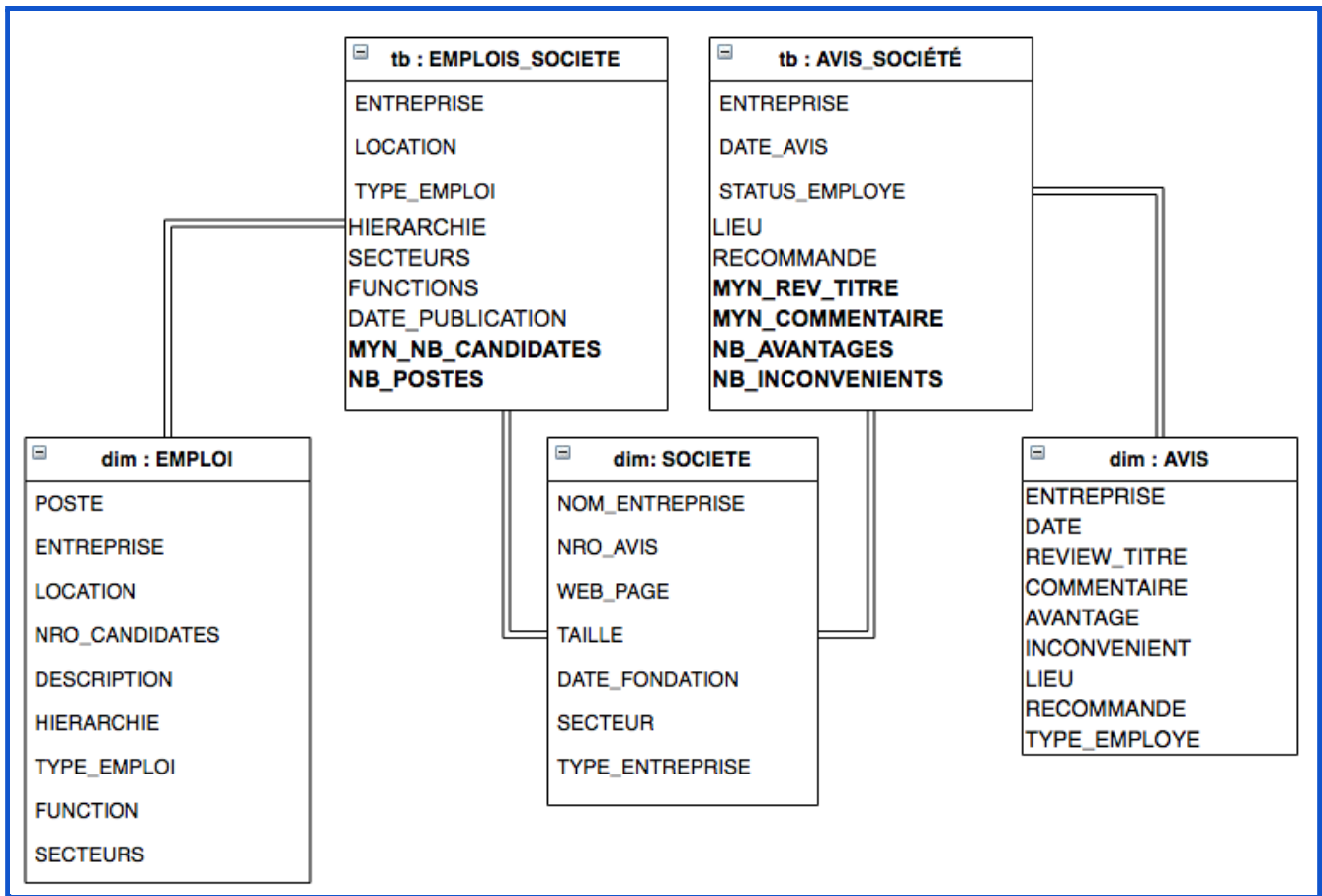
```

```

def Insert_Donnees_AVI() :
    con = psycopg2.connect(database="BASE_CURATED_ZONE", user="postgres", password="admin", host="127.0.0.1", port="5433")
    cur = con.cursor()
    myFilePathName = myPathRoot_CURATEDZONE + "AVI.txt"
    myFilePtr = open(myFilePathName, "r", encoding="utf-8", errors="ignore")
    myFileContents = myFilePtr.readlines()
    del myFileContents[0]
    for myLineRead in myFileContents:
        line = myLineRead.split(";")
        cle_unique = int(line[0])
        emplacement_source=line[1]
        datetime_ingestion=line[2]
        privacy_level=line[3]
        entreprise=line[4]
        date = line[5]
        review_titre=line[6]
        status_employe=line[7]
        lieu=line[8]
        recommande=line[9]
        commentaire=line[10]
        avantage= line[11].lower()
        inconvénient= line[12].lower()
        cur.execute("INSERT INTO AVIS VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
            (cle_unique,emplacement_source,datetime_ingestion,privacy_level,entreprise,date,review_titre,status_
employe,lieu,recommande,commentaire,avantage,inconvénient))
    myFilePtr.close()
con.commit()
con.close()
return (True)

```


A continuation le modèle de datawarehouse que nous avons construit :



Les mesures de la tf_EMPLOIS_SOCIETE correspondent au **nombre moyen de candidatures (MYN_NB_CANDIDATES)** et **nombre de postes publiés (NB_POSTES)** selon un poste publié, entreprise, location, hiérarchie et date publication.

```

def Insert_Donnees_FAIT_EMPLOIS() :
    con = psycopg2.connect(database="BASE_CURATED_ZONE", user="postgres", password="admin", host="127.0.0.1", port="5433")
    cur = con.cursor()
    cur.execute("select entreprise,location,type_emploi,hierarchie, secteurs, function, date_publication, count(poste) as
    nb_postes, round(avg(nro_candidates)) myn_nb_candidates from emploi group by entreprise,location,type_emploi,hierarchie,
    secteurs, function, date_publication order by entreprise,location,type_emploi,hierarchie, secteurs, function, date_publication")
    rows = cur.fetchall()
    for row in rows:
        cur.execute("INSERT INTO FAIT_EMPLOIS VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
            (row[0],row[1],row[2],row[3],row[4],row[5],row[6],row[7],row[8]))
    con.commit()
    cur.close()
    con.close()
    return (True)
  
```

Les mesures de la tf_AVIS_SOCIETE correspondent au **sentiment moyen dans les titres d'avis (MYN_REV_TITRE)**, **sentiment moyen dans le commentaire (MYN_COMMENTAIRE)**, **nombre**

d'avantages (AVANTAGES) et le nombre d'inconvénients (INCONVÉNIENTS) selon un entreprise, statut d'employé, lieu et date publication.

```
def Insert_Donnees_FAIT_AVIS() :
    con = psycopg2.connect(database="BASE_CURATED_ZONE", user="postgres", password="admin", host="127.0.0.1", port="5433")
    cur = con.cursor()
    cur.execute("select entreprise,date_avis,status_employe,lieu, recommande, avg(review_titre) as
myn_rev_titre,avg(commentaire) as myn_commentaire, count(avantage) avantages, count(inconvenient) inconvenients from
avis group by entreprise,date_avis,status_employe,lieu,recommande order by lieu")
    rows = cur.fetchall()
    for row in rows:
        cur.execute("INSERT INTO FAIT_AVIS VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
            (row[0],row[1],row[2],row[3],row[4],row[5],row[6],row[7],row[8],row[9]))
    con.commit()
    cur.close()
    con.close()
    return (True)
```

3.4 DATAVIZ

Pour le projet nous avons créé les suivants pages :

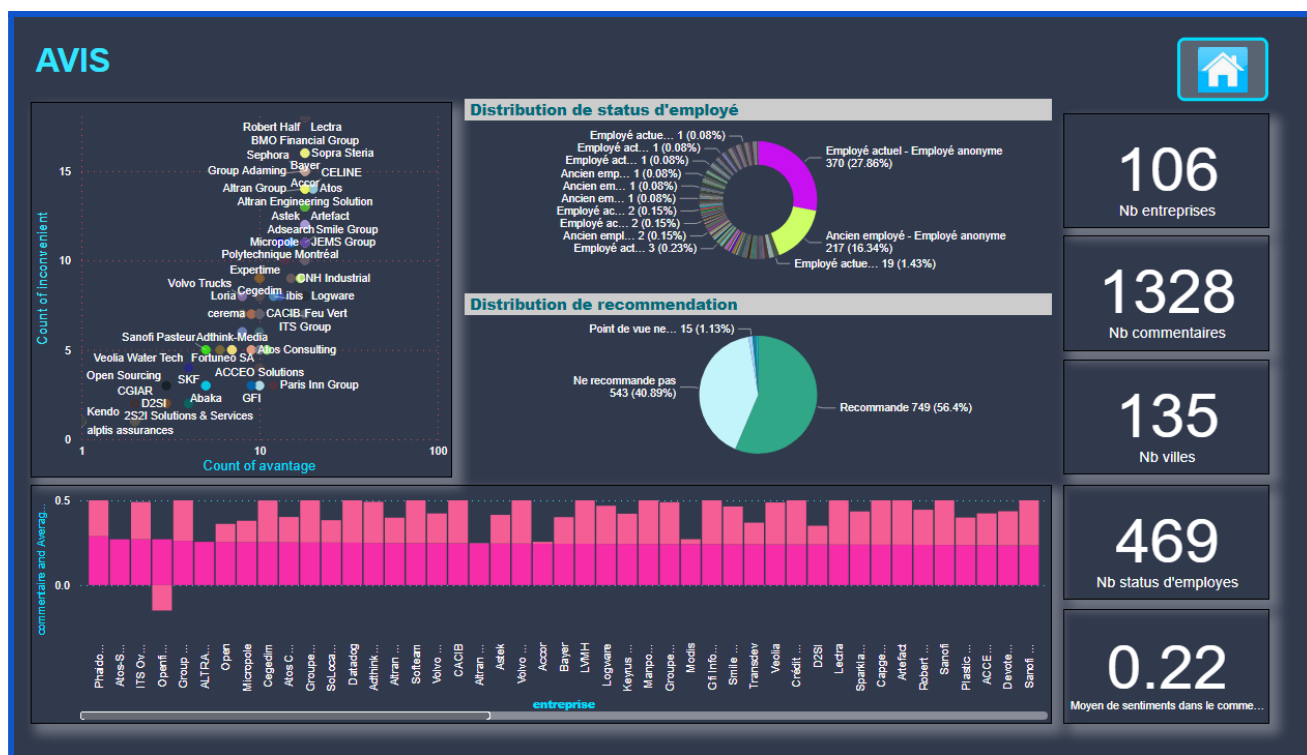
- **Avis,**
- **Sociétés,**
- **Emplois,**
- **Dashboard Final**

La page **Avis** concerne le fichier "avis.txt". Cette page présente 5 indicateurs : Nombre d'entreprises (nommées dans les avis), nombre de commentaires, nombre de villes, nombre distinct de statut d'employées et sentiment moyen dans les commentaires (valeur positive représente une opinion positive).

Il y a aussi 2 pie charts qui montrent la distribution du statut d'employé avec 27.86% pour statut : Employé actuel, et la distribution en la recommandation de l'entreprise avec 56,4% qui recommande positivement leur entreprise.

Dans la figure de nuage de points, nous trouvons la quantité d'avantages et inconvénients que offre chaque entreprise.

Finalement dans l'histogramme, nous pouvons observer le sentiment moyen pour le commentaire et pour le titre de chaque avis pour chaque entreprise.



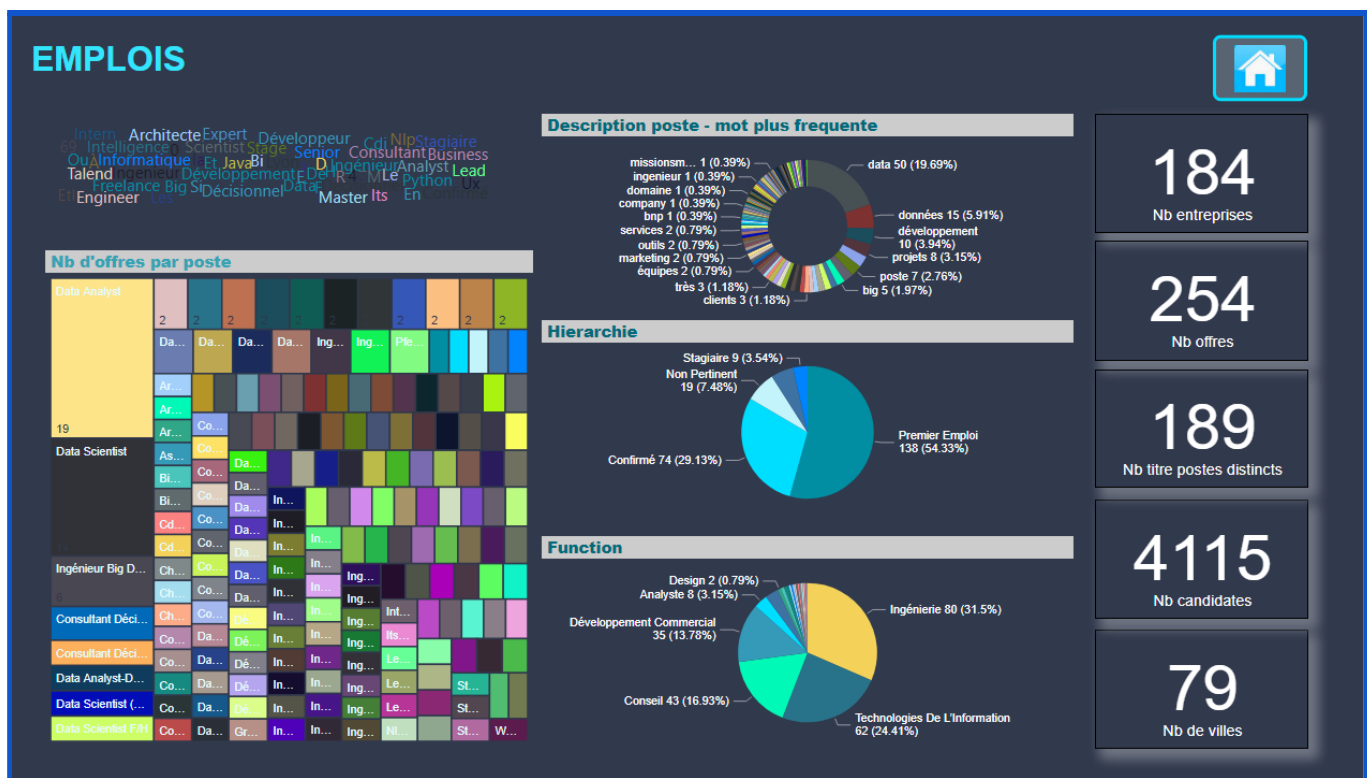
La page **EMPLOIS** concerne le fichier “emplois.txt”. Cette page présente 5 indicateurs : Nombre d’entreprises qui ont publié un poste, nombre d’offres, nombre de postes différents, nombre de candidats et nombre de villes.

Il y a aussi 2 pie charts qui montrent la distribution d’hiérarchie dans les postes avec 54.33% pour Premier emploi, et la distribution de fonctions du poste avec 31,5% pour l’ingénierie.

Dans la donut, nous trouvons la distribution pour le mot plus fréquente dans la description de chaque poste. Ici est le mot “DATA” avec 19,69%.

Dans la figure de nuage de mots, nous trouvons les postes plus demandés.

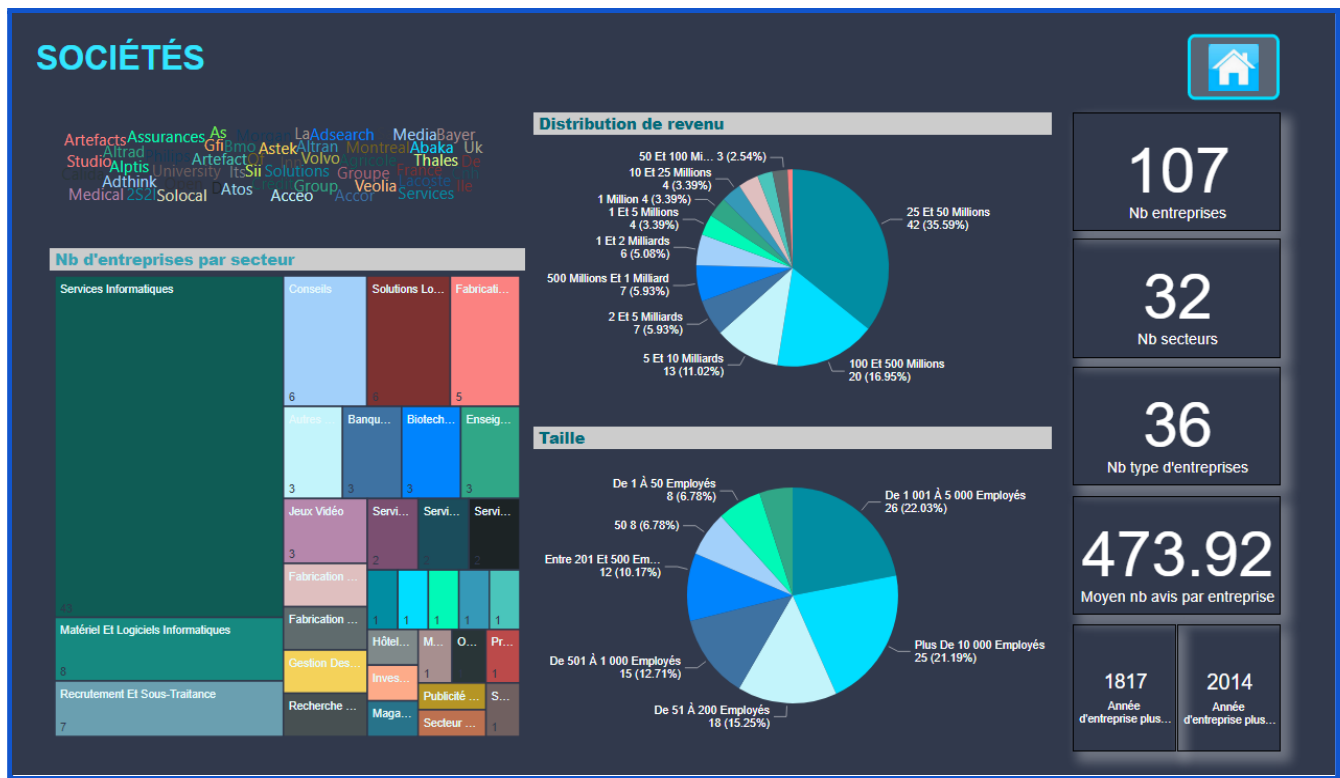
Finalement dans le treemap, nous pouvons observer le nombre d’offres par poste.



La page **SOCIETE** concerne le fichier “societe.txt”. Cette page présente 5 indicateurs : Nombre d’entreprises, nombre de secteurs, nombre de types d’entreprise différents, nombre moyen d’avis pour chaque entreprise et les années de fondation plus ancienne et récente d’entreprises.

Il y a aussi 2 pie charts qui montrent la distribution de revenus dans les postes avec 35.59% pour 25 et 50 Millions d’euros, et la distribution de taille avec 22,03% pour 1001 et 5000 employés.

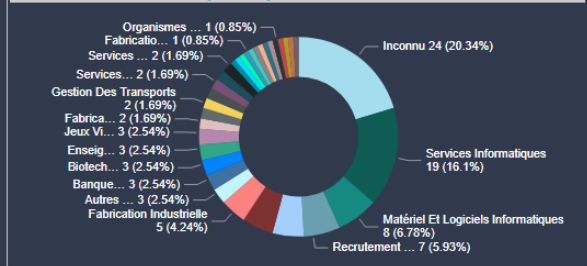
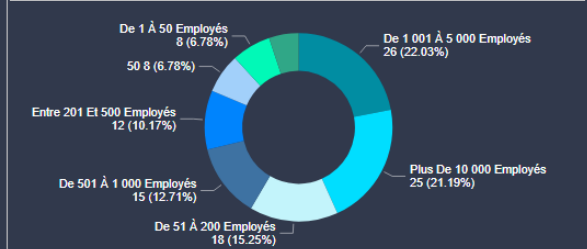
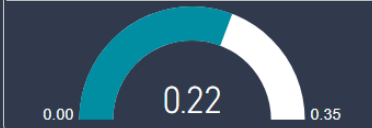
Finalement dans le treemap, nous pouvons observer le nombre d 'entreprises par secteur.



Dans le dashboard final, nous trouvons :

- Carte avec le nombre d'offres par ville
- Sentiment moyen dans chaque commentaire par société
- Distribution d'entreprises par taille
- Distribution d'entreprises par secteurs
- Nuage de postes plus fréquentes
- Nuage d'entreprise plus grands en taille
- 3 boutons qui permettent d'accéder à chacune des pages : **Avis, Emplois et Société**

BOUAZIZI M., CASANOVA S., CONDE K., RAMOS Y.



Titre	Type	Description
Datalake-Rapport	PDF	Rapport du projet
Lisez-moi	TXT	Instructions pour exécuter le code, charger la bd et observer la dataviz
Dataviz-PowerBI	PBIX et PDF	Dataviz
datalake-postgres	DUMP	Dump du projet, tables AVIS, EMPLOI, SOCIETE, TF_AVIS, TF_EMPLOIS
Variés: Résultats générés en chaque étape en py	TXT, CSV	Resultats plusieurs étapes du projet. Stockés dans le dossier Datalake-Projet-BO_CA_CO_RA
Variés : Fichier py projet	PY	Code réalisé en python pour le projet : Datalake_Programme_Principal, Datalake_Acquisition_des_donnes, Datalake_Extraction_Metadonnees, Datalake_Creation_Entrepot_Donnees. Stockés dans le

		dossier Datalake-Projet-BO_CA_CO_RA dossier Datalake-Projet-BO_CA_CO_RA
Lancer_Mise_En_Place_Datalake	BAT	Script en bash pour lancer le projet python. Stockés dans le dossier Datalake-Projet-BO_CA_CO_RA

REFERENCES

Mars 2021 :

1. <https://soat.developpez.com/tutoriels/bigdata/datalakes-architecture-big-data/>
2. <https://www.postgresqltutorial.com/postgresql-python/insert/>
3. <https://www.psycopg.org/docs/usage.html>
4. <https://blog.lesjeudis.com/web-scraping-avec-python>
5. <https://textblob.readthedocs.io/en/dev/quickstart.html>