VISUALITION DE DONNEES

Cas des données des effectifs des étudiants des années 2021/2022

EPSI

**Introduction**

Le domaine de l'éducation supérieure est en constante évolution, et la gestion efficace des effectifs d'étudiants devient cruciale pour les établissements. Ce projet vise à fournir une solution intégrée pour la collecte, la transformation, et l'analyse des données relatives aux effectifs d'étudiants inscrits dans les établissements d'enseignement supérieur. En utilisant des outils modernes tels que AWS S3, Snowflake et Apache Airflow, nous avons établi un processus robuste d'ingestion, de transformation, et de stockage des données, offrant une base solide pour des analyses approfondies.

Le rapport suivant détaille chaque phase du projet, de l'extraction initiale des données à la visualisation finale des résultats. Nous explorerons également les tendances clés observées dans les effectifs étudiants, ainsi que les avantages découlant de l'orchestration automatisée à l'aide d'Airflow et de l'utilisation de Snowpipe pour l'ingestion continue des données dans Snowflake.

Ce rapport est structuré pour fournir une compréhension approfondie du processus de bout en bout, en mettant en évidence les principaux points forts des visualisations créées ainsi que les perspectives d'amélioration future. En explorant chaque étape du projet, nous espérons fournir une ressource complète pour les parties prenantes qui cherchent à mieux comprendre l'évolution des effectifs étudiants dans l'enseignement supérieur.

1. **Extraction des Données**

La phase d'extraction des données est une étape cruciale dans le processus global du projet. Pour cette tâche, nous avons utilisé la bibliothèque pandas en Python pour lire les données depuis l'URL fournie par le lien de téléchargement.

La fonction extract\_data prend en charge cette opération. Voici la mise en œuvre de la fonction :

def extract\_data(\*\*kwargs):

    url=kwargs['url']

    df=pd.read\_csv(url, sep=';')

    now=datetime.now()

    date\_now\_string= now.strftime("%d%m%Y%H%M%S")

    file\_str= 'data\_gouv\_'+date\_now\_string

    df.to\_csv(f"{file\_str}.csv", index=False, sep=';')

    output\_file\_path=f"/home/lifu237/{file\_str}.csv"

    output\_list= [output\_file\_path,file\_str]

    return output\_list

Cette fonction prend l'URL en paramètre, utilise pandas pour lire les données CSV, puis les enregistre dans un nouveau fichier CSV en utilisant le séparateur ';' pour garantir la cohérence des données. Le fichier résultant est enregistré avec un nom unique basé sur le timestamp actuel. Enfin, la fonction retourne le chemin du fichier résultant ainsi que le nom du fichier pour une utilisation ultérieure dans le pipeline.

1. **Transformation des Données et chargement des données**

La phase de transformation des données est importante pour préparer les données brutes en vue de leur utilisation ultérieure. Dans cette partie du pipeline, nous appliquons diverses opérations pour nettoyer, formater et enrichir les données extraites.

def transform\_data(task\_instance):

    data=task\_instance.xcom\_pull(task\_ids="tsk\_extract\_data\_gouv")[0]

    object\_key= task\_instance.xcom\_pull(task\_ids="tsk\_extract\_data\_gouv")[1]

    df=pd.read\_csv(data, sep=';')

    print("Columns in DataFrame:")

    print(df.columns)

    print("\nFirst few rows of DataFrame:")

    print(df.head())

    columns=['rentree', 'categorie\_etablissement','secteur\_etablissement','sigle\_etablissement', 'libelle\_etablissement\_1','libelle\_etablissement\_2','reg\_nom','aca\_nom','geo','degetu','degre\_etudes','effectifhdccpge',

         'dont\_femmes','dont\_hommes']

    df=df[columns]

    df.drop\_duplicates()

    df[['latitude', 'longitude']] = df['geo'].str.extract(r'([0-9]+\.[0-9]+),\s\*([0-9]+\.[0-9]+)', expand=True)

    df['latitude'] = pd.to\_numeric(df['latitude'])

    df['longitude'] = pd.to\_numeric(df['longitude'])

    df = df.drop('geo', axis=1)

    df['effectifhdccpge'] = pd.to\_numeric(df['effectifhdccpge'], errors='coerce')

    df = df.dropna(subset=['effectifhdccpge'])

    csv\_data=df.to\_csv(index=False, sep=';')

    print('Num of rows:', len(df))

    print('Num of cols:', len(df.columns))

Cette fonction effectue plusieurs étapes de transformation, notamment la sélection des colonnes pertinentes, l'extraction des coordonnées géographiques (latitude et longitude) à partir de la colonne 'geo', la conversion des types de données, et la suppression des valeurs manquantes dans la colonne 'effectifhdccpge'.

La phase de chargement consiste à transférer les données transformées dans un emplacement spécifié, généralement un stockage centralisé. Les données transformées sont ensuite enregistrées dans un fichier CSV, et le fichier est chargé dans un nouveau bucket S3 via la commande suivante :

 s3\_client.put\_object(Bucket=target\_bucket\_name, Key=object\_key, Body=csv\_data)

Par ailleurs, nous chargeons aussi la donnée brute dans un autre bucket S3 pour des soucis éventuels de backup. Pour ce faire nous utilisons la commande Bash suivante :

load\_to\_s3=BashOperator(

        task\_id='tsk\_load\_to\_s3',

        bash\_command='aws s3 mv {{ ti.xcom\_pull("tsk\_extract\_data\_gouv")[0] }} s3://olivier-datagouv-education-raw'

    )

1. **Stockage Snowflake et ingestion snowpipe**

Cette étape consiste à charger les données transformées dans notre base de données Snowflake. Nous avons créé une table dans Snowflake avec les colonnes appropriées et défini un format de fichier pour interagir avec le CSV. En utilisant Snowpipe, nous avons automatisé le processus d'ingestion continue des données depuis notre zone de stockage externe (S3) vers Snowflake. Cette approche garantit une synchronisation constante entre nos données source et notre datawarehouse, réduisant ainsi les retards potentiels.

Snowpipe offre une gestion automatisée des mises à jour, s'assurant que notre entrepôt Snowflake reste constamment à jour avec les dernières données disponibles. Le pipe Snowpipe surveille notre zone de stockage externe et déclenche automatiquement le chargement des nouvelles données dès qu'elles deviennent disponibles.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Cette approche intégrée d'ingénierie des données garantit que notre entrepôt Snowflake est continuellement alimenté en données éducatives à jour, prêtes à être exploitées pour des analyses approfondies. Cela nous permet de prendre des décisions éclairées basées sur des informations actualisées, tout en automatisant le processus pour gagner en efficacité opérationnelle.

1. **Orchestration avec Apache Airflow**

L'orchestration de notre pipeline de données éducatives est gérée de manière transparente et efficace par Apache Airflow, un système open-source pour l'orchestration de workflows. Voici comment notre orchestration fonctionne dans le contexte de ce projet. Dans Airflow, nous avons créé un DAG nommé 'data\_gouv\_analytics\_dag' qui définit l'ensemble du flux de travail. Ce DAG est configuré pour s'exécuter mensuellement, garantissant la régularité des mises à jour de nos données éducatives.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Airflow gère la planification régulière du DAG, s'assurant que notre pipeline s'exécute de manière fiable et prévisible. En cas d'échec d'une tâche, Airflow offre des mécanismes intégrés pour la gestion des erreurs, la reprise et les notifications, garantissant la robustesse et la surveillance de notre flux de travail. Cette orchestration automatisée garantit la cohérence et la fiabilité de notre flux de données, tout en nous offrant une visibilité complète sur le statut et l'exécution de chaque tâche, contribuant ainsi à la robustesse et à l'efficacité de notre pipeline.

1. **Visualisations des données**

Durant notre analyse, nous avons fait ressortir plusieurs graphiques tels que :

Le nombre d’étudiants total, d’hommes et de femme, sur les deux années et même sur chaque année.

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

La région qui accueille le plus d’étudiants

Une image contenant texte, capture d’écran, logiciel, ordinateur

Description générée automatiquement

Le taux d’étudiants accueillis par les différents secteurs privé et public et on peut noter les variations de population durant chaque année du lycée au bac +6

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Il y’a une variété de graphiques qui peuvent être exploités pour comprendre la population estudiantine par le secteur, les catégories d’établissement ou encore le sexe.

1. **Perspectives d'Amélioration**

Après la production de notre tableau de bord, nous avons noté qu’il y’a de potentielles améliorations à faire pour rentrer plus en profondeur dans cette thématique. Nous parlerons de deux d’entre elles.

* Enrichissement des Données :

Explorer la possibilité d'enrichir les données existantes avec des sources complémentaires. Par exemple, l'ajout d'informations socio-économiques des régions académiques ou l'intégration de données démographiques supplémentaires pourrait offrir une perspective plus complète.

* Développement de Modèles de Prédiction Avancés :

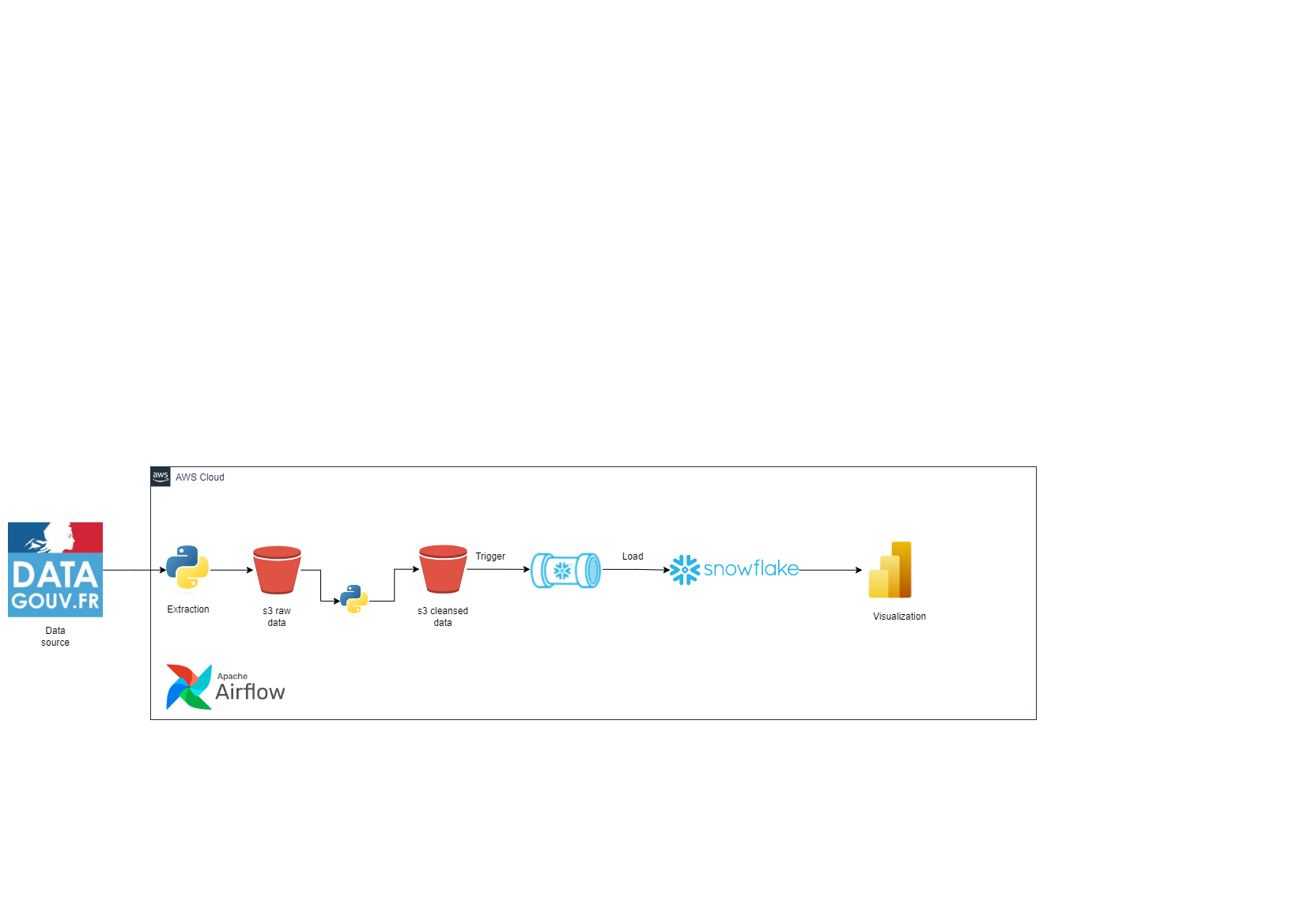
Si la prédiction des tendances éducatives est une priorité, envisager le développement de modèles de prédiction plus avancés. Cela pourrait inclure l'utilisation d'algorithmes d'apprentissage profond ou d'autres techniques d'analyse prédictive.

**Conclusion**

En conclusion, ce projet offre une solution complète et évolutive pour l'analyse des effectifs étudiants dans l'enseignement supérieur. En combinant les meilleures pratiques d'ingénierie de données, d'orchestration et de visualisation, le pipeline répond aux besoins actuels tout en étant prêt à évoluer pour faire face aux futurs défis du secteur éducatif. Il constitue un outil puissant pour les décideurs et les responsables politiques cherchant à améliorer la qualité et la pertinence de l'enseignement supérieur.

**Annexes**

Architecture du projet



Datagouv\_analytics.py

from airflow import DAG

from datetime import timedelta,datetime

from airflow.operators.python import PythonOperator

import pandas as pd

import boto3

from airflow.operators.bash import BashOperator

s3\_client=boto3.client('s3')

target\_bucket\_name='olivier-datagouv-education-transform'

url\_data='https://www.data.gouv.fr/fr/datasets/r/b10fd6c8-6bc9-41fc-bdfd-e0ac898c674a'

def extract\_data(\*\*kwargs):

    url=kwargs['url']

    df=pd.read\_csv(url, sep=';')

    now=datetime.now()

    date\_now\_string= now.strftime("%d%m%Y%H%M%S")

    file\_str= 'data\_gouv\_'+date\_now\_string

    df.to\_csv(f"{file\_str}.csv", index=False, sep=';')

    output\_file\_path=f"/home/lifu237/{file\_str}.csv"

    output\_list= [output\_file\_path,file\_str]

    return output\_list

def transform\_data(task\_instance):

    data=task\_instance.xcom\_pull(task\_ids="tsk\_extract\_data\_gouv")[0]

    object\_key= task\_instance.xcom\_pull(task\_ids="tsk\_extract\_data\_gouv")[1]

    df=pd.read\_csv(data, sep=';')

    print("Columns in DataFrame:")

    print(df.columns)

    print("\nFirst few rows of DataFrame:")

    print(df.head())

    columns=['rentree', 'categorie\_etablissement','secteur\_etablissement','sigle\_etablissement', 'libelle\_etablissement\_1','libelle\_etablissement\_2','reg\_nom','aca\_nom','geo','degetu','degre\_etudes','effectifhdccpge',

         'dont\_femmes','dont\_hommes']

    df=df[columns]

    df.drop\_duplicates()

    df[['latitude', 'longitude']] = df['geo'].str.extract(r'([0-9]+\.[0-9]+),\s\*([0-9]+\.[0-9]+)', expand=True)

    df['latitude'] = pd.to\_numeric(df['latitude'])

    df['longitude'] = pd.to\_numeric(df['longitude'])

    df = df.drop('geo', axis=1)

    df['effectifhdccpge'] = pd.to\_numeric(df['effectifhdccpge'], errors='coerce')

    df = df.dropna(subset=['effectifhdccpge'])

    csv\_data=df.to\_csv(index=False, sep=';')

    print('Num of rows:', len(df))

    print('Num of cols:', len(df.columns))

    object\_key=f"{object\_key}.csv"

    s3\_client.put\_object(Bucket=target\_bucket\_name, Key=object\_key, Body=csv\_data)

default\_args={

    'owner':'olivierassiene',

    'depends\_on\_past':False,

    'start\_date':datetime(2024,1,17),

    'email':'connextu.webtech@gmail.com',

    'email\_on\_failure':False,

     'email\_on\_retry':False,

    'retries':2,

    'retry\_delay':timedelta(seconds=30)

}

with DAG('data\_gouv\_analytics\_dag',

         default\_args=default\_args,

         schedule\_interval='@monthly',

         catchup=False) as dag:

    extract\_data\_gouv=PythonOperator(

        task\_id='tsk\_extract\_data\_gouv',

        python\_callable=extract\_data,

        op\_kwargs={'url':url\_data}

    )

    transform\_data\_gouv=PythonOperator(

        task\_id='tsk\_transform\_data\_gouv',

        python\_callable=transform\_data

    )

    load\_to\_s3=BashOperator(

        task\_id='tsk\_load\_to\_s3',

        bash\_command='aws s3 mv {{ ti.xcom\_pull("tsk\_extract\_data\_gouv")[0] }} s3://olivier-datagouv-education-raw'

    )

extract\_data\_gouv>>transform\_data\_gouv>>load\_to\_s3

Script\_snowflake

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement