INSEA

Data Warehouse & Business Analytics

Système décisionnel de gestion d'absence

-Rapport-

Encadré par : **Pr. Imade BENELALLAM**

Réalisé par :
- ENNAJI AYOUB
- ELHOURI MOHAMED EL MEHDI

Filière : DSE

Semestre : S4

TABLE DES MATIERES

LISTE DES FIGURES

INTRODUCTION

Le corps professoral et administratif souhaite controller et suivre l'assiduité des étudiants lors de cette phase d'enseignement à distance. D'où l'idée de réaliser un Système décisionnel de gestion d'absence.

Ce projet s'inscrit dans le cadre de l'élément de module Data Warehouse & Business Analytics. Il est réalisé sur cloud en utilisant Airflow, Pandas, pandasql, Os et shutil.

I- Dossier jupyter

Jupyter Notebook permet aux développeurs de partager du code et de l'exécuter dans la même interface utilisateur. Il peut associer du code, des graphiques, des visualisations et du texte dans des notebooks - ou cahiers - partageables qui s'exécutent dans un navigateur web.

Figure 1: data_ folder de jupyter

Il contient 3 dossiers :

- Input : ou on ajoute les fichiers à traiter
- Output : contient le résultat du traitement
- Processed : contient des copies des fichiers existant dans le dossier input après traitement.

II- Visualisation du Dag

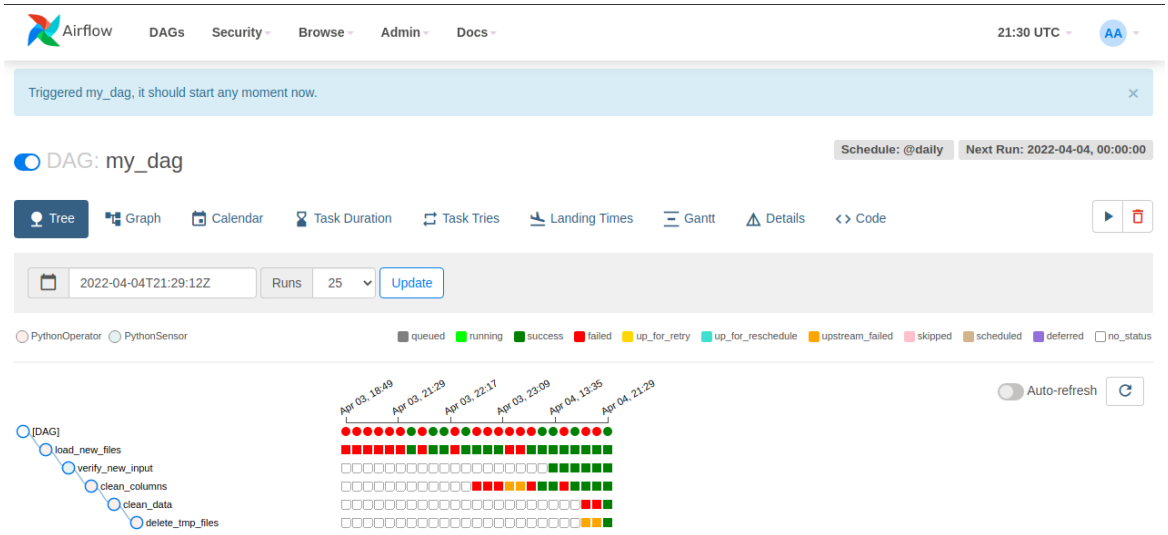

Figure 2: Visualisation des Dags 1

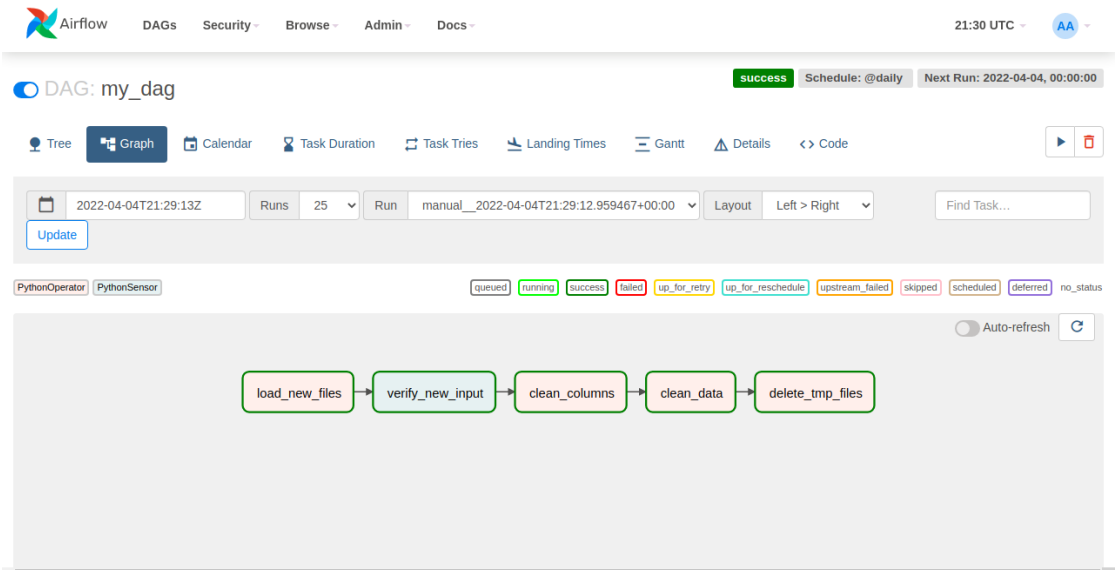

Figure 3: visualisation des Dags 2

On remarque la présence de 5 tâches qui sont :

1- Load_new_files : permet d'importer des nouveaux fichiers depuis le dossier input ;

```python
def _load_new_files(ti):
    dfs = []
    id = "";
    for filename in os.listdir(INPUT):
        if filename not in os.listdir(PROCESSED):
            df = pd.read_csv(INPUT +'/'+ filename, skiprows=7,
                             delimiter="\t", encoding="UTF-16")
            df['Meeting_Id'] = _get_meeting_id(INPUT +'/'+ filename)
            dfs.append(df)
            shutil.copyfile(INPUT +'/'+ filename, PROCESSED +'/'+ filename)
    if len(dfs) == 0:
        ti.xcom_push(key='new_files_found', value=False)
        return

    result = pd.concat(dfs)
    os.mkdir(TMP + '/first/')
    os.chdir(TMP + '/first/')
    result.to_csv(str( len(os.listdir(TMP + '/first/')) + 1)
                  + '.csv', encoding="UTF-16")
    ti.xcom_push(key='new_files_found', value=True)
```

Figure 4: fonction « load_new_file »

2- Verify_new_input : vérifie si il y'a des nouveaux fichiers non-traités.

```python
def _verify_new_input(ti):
    return ti.xcom_pull(key='new_files_found',
           task_ids=['_load_new_files'])
```

Figure 5: fonction « verify_new_input »

3- Clean_columns : permet de modifier les noms des colonnes, la forme, ajouter ou supprimer des champs…

```python
def _clean_columns():
    df = pd.read_csv(TMP +'/first/'+ str( len(os.listdir(TMP + '/first/')))
                    + '.csv', encoding="UTF-16")

    df.drop('Adresse de courrier',axis=1, inplace=True)

    df.columns=['ligne','nom_participant','h_arr', 'h_dep','duree_en_mins',
                'role','id_participant', 'id_meeting']
    df = df[['id_meeting', 'h_arr', 'h_dep', 'duree_en_mins', 'id_participant',
             'nom_participant','role']]

    raw_ha = [datetime.strptime(i, '%m/%d/%Y, %H:%M:%S %p') for i in df['h_arr']]
    raw_hd = [datetime.strptime(i, '%m/%d/%Y, %H:%M:%S %p') for i in df['h_dep']]

    df['meeting_date'] = [datetime.strftime(i, '%d/%m/%Y') for i in raw_ha]
    df['duree_en_mins'] = [ int((i - j).total_seconds()//60)
                           for i, j in zip(raw_hd, raw_ha)]
    df['h_arr'] = [datetime.strftime(i,'%H:%M:%S') for i in raw_ha]
    df['h_dep'] = [datetime.strftime(i,'%H:%M:%S') for i in raw_hd]
    df['role'] = [i[0] for i in df['role']]

    os.mkdir(TMP + '/second/')
    os.chdir(TMP + '/second/')
    df.to_csv(str( len(os.listdir(TMP + '/second/')) + 1) + '.csv',
              encoding="UTF-16")
```

Figure 6: fonction « Clean_culumns »

4- Clean_data : permet de créer des tables intermédiaires dont le but d'avoir une table finale contenant les données souhaitées.

```
def _clean_data():

    df = pd.read_csv(TMP +'/second/'+ str( len(os.listdir(TMP + '/second/')))
                     + '.csv', encoding="UTF-16")

    query = '''SELECT id_meeting,meeting_date ,id_participant, nom_participant, role,
    min(h_arr) AS premiere_arrivee,
    max(h_dep) AS dernier_depart, SUM(duree_en_mins) AS duree_totale_presence
    from df
    GROUP BY id_meeting,meeting_date,id_participant, nom_participant, role
    '''
    df2 = sqldf(query,locals())

    query2 = '''SELECT id_meeting, id_participant AS id_organisateur,
    nom_participant AS nom_organisateur,
    premiere_arrivee AS arrivee_organisateur,
    dernier_depart AS depart_organisateur
    FROM df2 WHERE role = 'O'
    '''
    df3 = sqldf(query2,locals())

    df4 = pd.merge(df2, df3, how='inner', on='id_meeting')

    df5 = df4.drop('role',axis=1)

    pa = [datetime.strptime(i,'%H:%M:%S') for i in df5['premiere_arrivee']]
    ao = [datetime.strptime(i,'%H:%M:%S') for i in df5['arrivee_organisateur']]

    df5['retard_en_mins']= [int((i - j).total_seconds()//60)
                            if (i-j).total_seconds() >= 0 else 0 for i,j in zip(pa,ao)]

    dd = [datetime.strptime(i,'%H:%M:%S') for i in df5['dernier_depart']]
    do = [datetime.strptime(i,'%H:%M:%S') for i in df5['depart_organisateur']]

    df5['exces_en_mins']= [int((i - j).total_seconds()//60)
                            if (i-j).total_seconds() >= 0 else 0 for i,j in zip(dd,do)]

    df5['duree_reelle_presence'] = [i -j -k if i-j-k >= 0 else 0 for i,j,k in
zip(df5['duree_totale_presence'],df5['exces_en_mins'],df5['retard_en_mins'])]

    df5['duree_meeting_en_mins'] = [int((i - j).total_seconds()//60) for i,j in zip(do,ao)]
    df5 = df5[['id_meeting', 'meeting_date','duree_meeting_en_mins', 'id_organisateur'
,'nom_organisateur','id_participant', 'nom_participant', 'retard_en_mins',
                'duree_reelle_presence']]

    os.chdir(OUTPUT)
    df5.to_csv(str( len(os.listdir(OUTPUT)) + 1) + '.csv', encoding="UTF-16")

```

Figure 7: fonction : « Clean_data »

5- Delete_tmp_file : elle supprime les fichiers temporaires.

```python
def _delete_tmp_files():
    shutil.rmtree(TMP)

with DAG("my_dag", start_date=datetime(2022, 4, 3), schedule_interval="@daily",
         catchup=False ) as dag:

    load_new_files = PythonOperator(
        task_id = "load_new_files",
        python_callable = _load_new_files
    )

    verify_new_input = PythonSensor(
        task_id = "verify_new_input",
        python_callable = _verify_new_input,
        soft_fail = True
    )

    clean_columns = PythonOperator(
        task_id = "clean_columns",
        python_callable = _clean_columns
    )

    clean_data = PythonOperator(
        task_id = "clean_data",
        python_callable = _clean_data
    )

    delete_tmp_files = PythonOperator(
        task_id = "delete_tmp_files",
        python_callable = _delete_tmp_files
    )

    load_new_files >> verify_new_input >> clean_columns >> clean_data >> delete_tmp_files
```

Figure 8: fonction « Delet_tmp_files »

CONCLUSION

La réalisation de ce travail a été très bénéfique et très enrichissante. Elle nous a permis d'utiliser des technologies nouvelles tel que Airflow, Pandas, pandasql…

Cette solution s'avère être très utile pour la gestion de l'absence. Nous allons l'améliorer au fur et à mesure de l'avancement du cours afin de produire une solution permettant d'aider le corps professoral et administratif à prendre les mesures nécessaires pour gérer assurer la présence.