

Connexion et extraction des données

Utilisation de **Microsoft SQL Server Management Studio (SSMS)** pour accéder aux fichiers, interroger les tables et extraire les informations pertinentes.

Pour l'installation de **Microsoft SQL Server Management Studio (SSMS)** j'ai suivi les tutos de [Anders Jensen](https://www.youtube.com/watch?v=7zXtA0LwoHs) via le lien : <https://www.youtube.com/watch?v=7zXtA0LwoHs>

Pour la connexion toujours choisir Facultatif au niveau de chiffrement comme sur le graphe

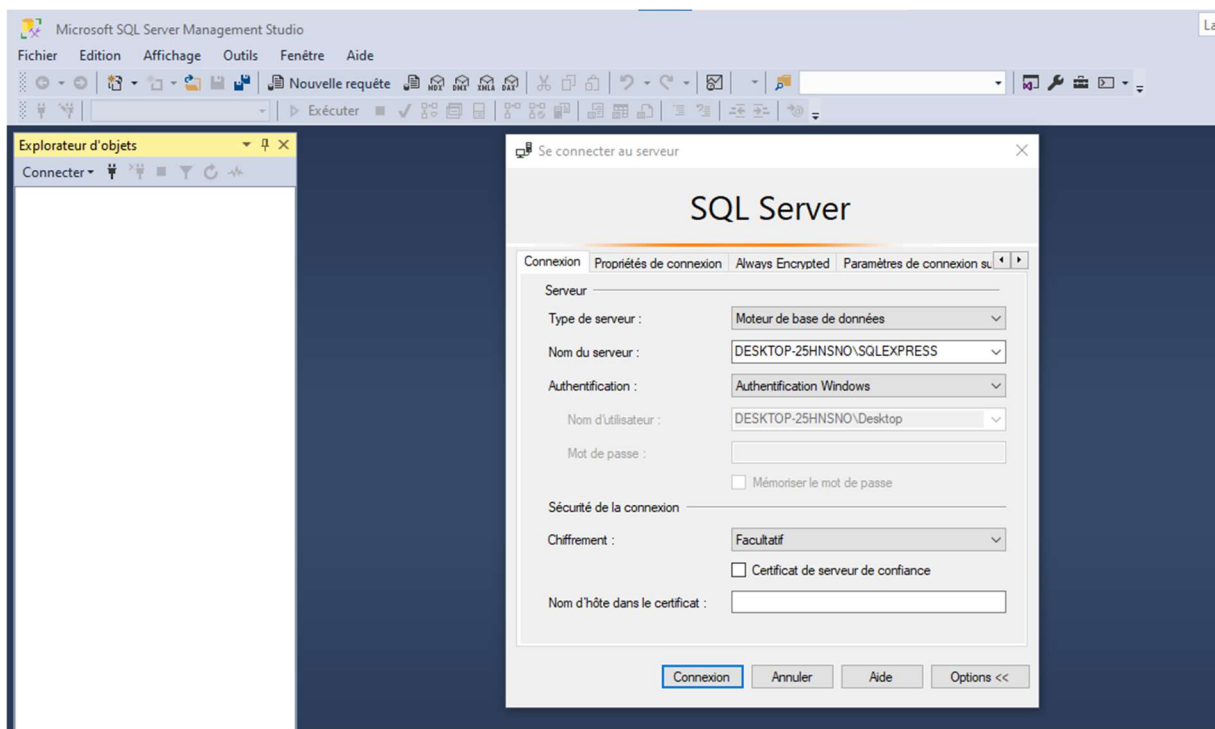


Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-1 : connexion SQL Server

Clic droit sur Base de données ➡ joindre

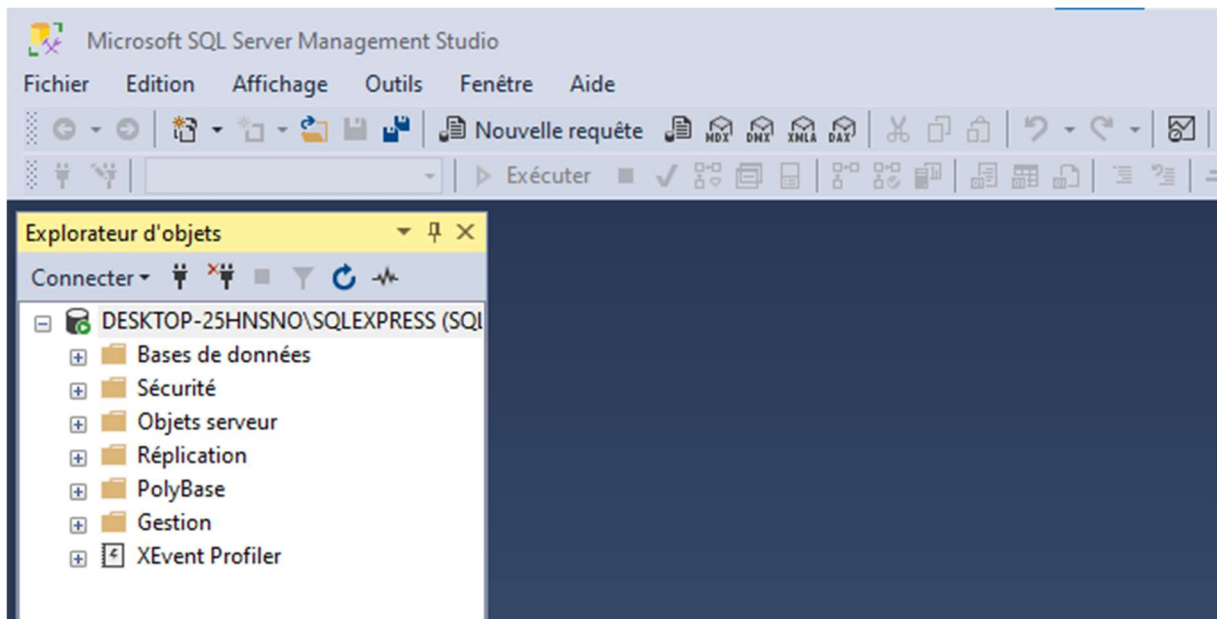


Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-2 : page d'accueil SSMS

Ajouter ➡ choisir le fichier .mdf ➡ choisir le fichier .ldf ➡ OK

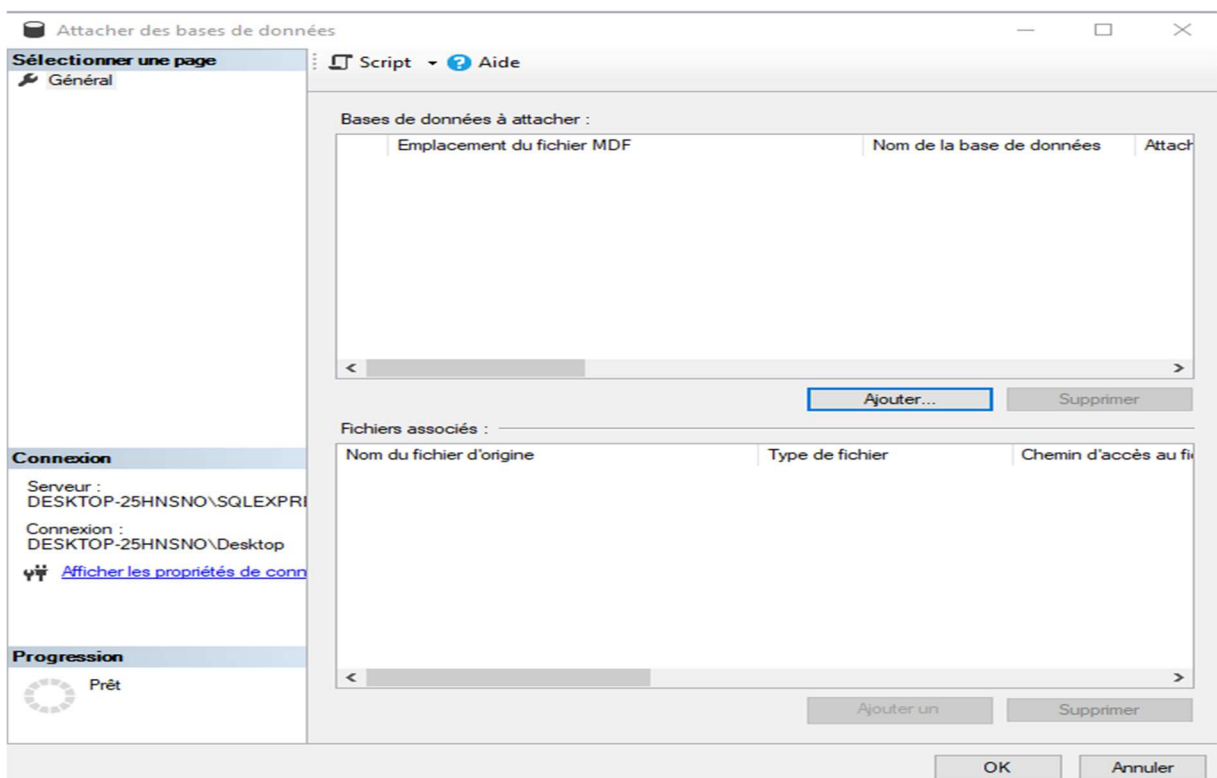


Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-3 : IMPORTATION DU BASE DE DONNEE

Les bases de données sont enfin importées comme le montre le graphe suivant

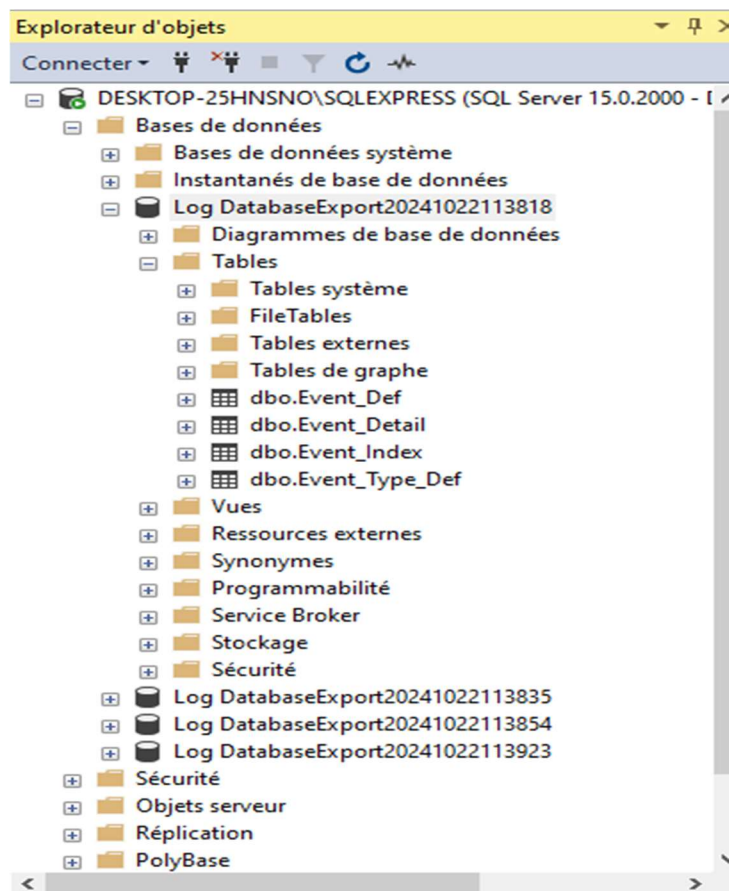


Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-4 : importation base de données réussis

Pour afficher les tables comme sur le graphe suivant il faut :

Clic droit sur la table qu'on veut afficher ➡ cliquer sur sélectionner les 1000 premiers lignes

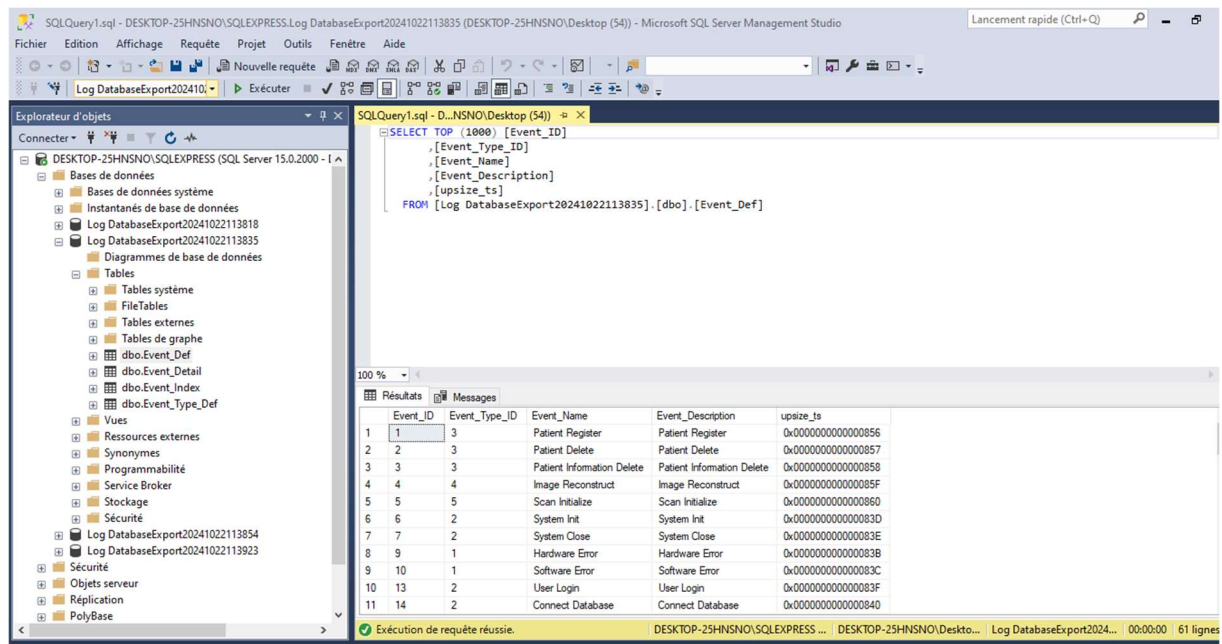


Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-5 : affichage table

Conversion en format exploitable

La conversion de nos données privées en un format exploitable a été effectuée entièrement avec **Python**. Le processus a consisté à établir la connexion au serveur SQL, à exporter les tables de la base de données, puis à les fusionner dans un fichier **.csv** afin de préparer les données pour les phases ultérieures d'analyse.

```
import pyodbc
import pandas as pd
import os

# Paramètres pour chaque base de données
server = r'DESKTOP-25HNSNO\SQLEXPRESS'
databases = ['Log DatabaseExport20241022113835', 'Log DatabaseExport20241022113854', 'Log DatabaseExport20241022113923']

# Dossier de sortie pour le CSV combiné
output_folder = r'C:\Users\Desktop\ExportCSV_MultiBase'
os.makedirs(output_folder, exist_ok=True)

# Liste pour stocker tous les DataFrames
all_dataframes = []
```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-6 : l'initialisation des variables

```

# ♦ Boucle sur chaque base
for db in databases:
    print(f"\n♦ Connexion à la base : {db}")
    conn_str = f'DRIVER={{SQL Server}};SERVER={server};DATABASE={db};Trusted_Connection=yes;'
    conn = pyodbc.connect(conn_str)

    # Récupérer toutes les tables de la base
    tables_query = """
    SELECT TABLE_SCHEMA, TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_TYPE='BASE TABLE';
    """
    tables = pd.read_sql(tables_query, conn)

    # Boucle sur chaque table
    for index, row in tables.iterrows():
        schema = row['TABLE_SCHEMA']
        table_name = row['TABLE_NAME']
        print(f" Export de la table : {schema}.{table_name}")

        query = f"SELECT * FROM [{schema}].[{table_name}]"
        df = pd.read_sql(query, conn)

        # Ajouter colonnes pour source
        df['Source_Base'] = db
        df['Source_Table'] = f"{schema}.{table_name}"

        all_dataframes.append(df)

```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-7 : récupération des données de l'ensemble des tables de chaque base

```

# ♦ Combiner toutes les tables en gérant les colonnes différentes
combined_df = pd.concat(all_dataframes, ignore_index=True, sort=False)

```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-8 : fusion des données

Nettoyage du fichier des données privées

```

# ♦ Nettoyage automatique (optionnel)
# Remplacer les valeurs manquantes par une valeur vide
combined_df.fillna('', inplace=True)

# ♦ Exporter Le CSV final
combined_csv = os.path.join(output_folder, 'fichier_fusionne.csv')
combined_df.to_csv(combined_csv, index=False, sep=';', encoding='utf-8')

print(f"\n✅ Toutes les bases ont été combinées avec succès dans : {fichier_fusionne.csv}")

```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-9 : nettoyage de fichier-fusionne.csv

Normalisation du fichier des données privés

```

import pandas as pd
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

# Copier le dataframe pour ne pas écraser l'original
df_encoded = df_test.copy()

# Colonnes catégorielles à encoder
cat_cols = ['Key_Name_f1', 'Key_Value_f1', 'Key_Name_f2', 'Key_Value_f2',
            'Event_Type_ID', 'Event_Name', 'Event_Type_Name', 'Access_Authority']

# Encoder Les colonnes catégorielles
le = LabelEncoder()
for col in cat_cols:
    df_encoded[col] = le.fit_transform(df_encoded[col].astype(str))

# Pour Event_Time, extraire heure, minute, seconde
df_encoded['Event_Time'] = pd.to_datetime(df_encoded['Event_Time'], errors='coerce')
df_encoded['Event_Hour'] = df_encoded['Event_Time'].dt.hour.fillna(0)
df_encoded['Event_Minute'] = df_encoded['Event_Time'].dt.minute.fillna(0)
df_encoded['Event_Second'] = df_encoded['Event_Time'].dt.second.fillna(0)
df_encoded.drop(columns=['Event_Time'], inplace=True)

# Identifier toutes les colonnes numériques pour normalisation
num_cols = df_encoded.select_dtypes(include=['int64', 'float64']).columns

# Normalisation
scaler = MinMaxScaler()
df_encoded[num_cols] = scaler.fit_transform(df_encoded[num_cols])

```

Figure 5-Erreur ! Il n'y a pas de texte répondant à ce style dans ce document.-10 : normalisation de fichier-fusionne.csv