Project
Data Warehousing

Supervised by : Mrs Wiem Zaouga

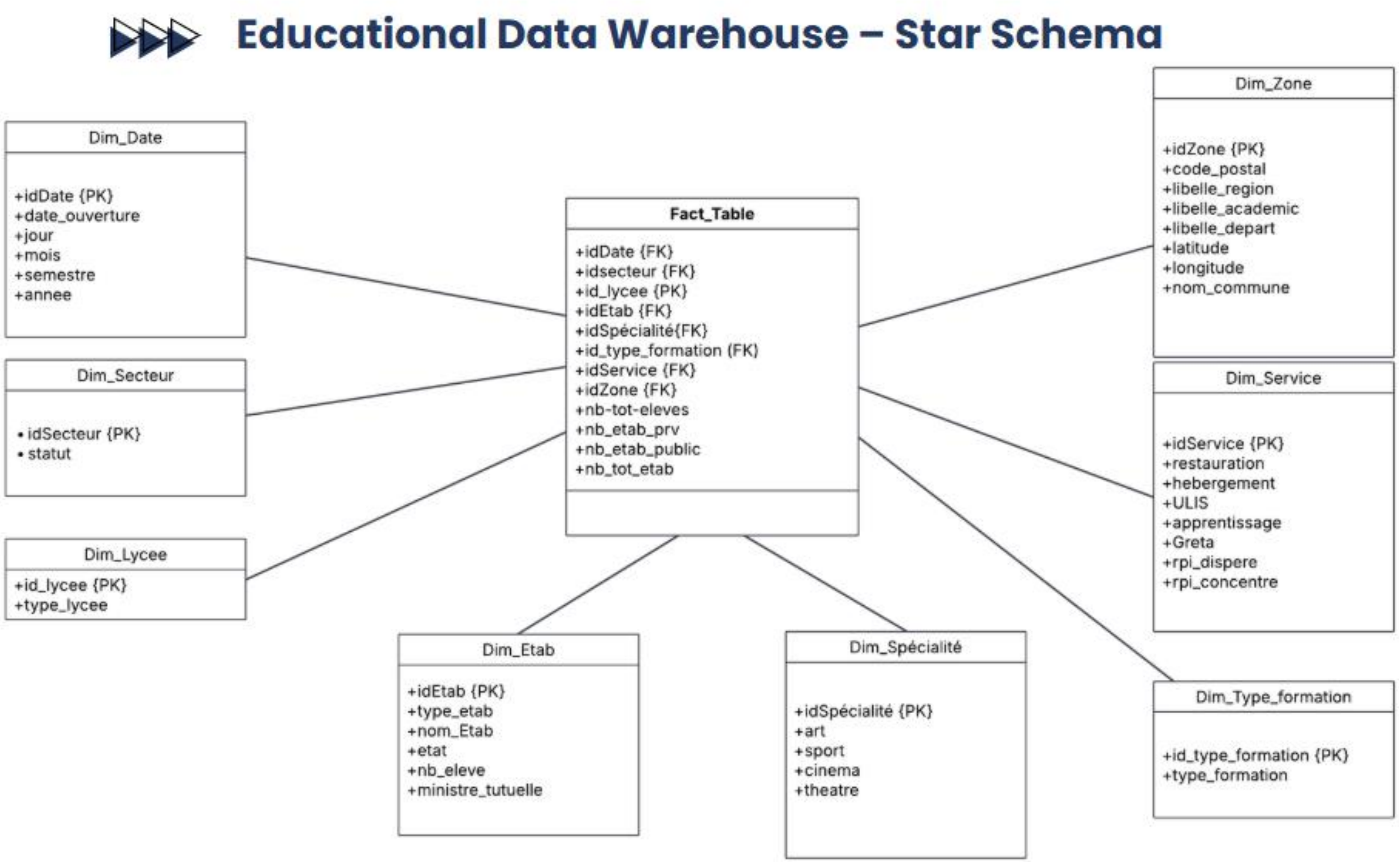
Mrs Ines Slimen

Summary

1-General Context :

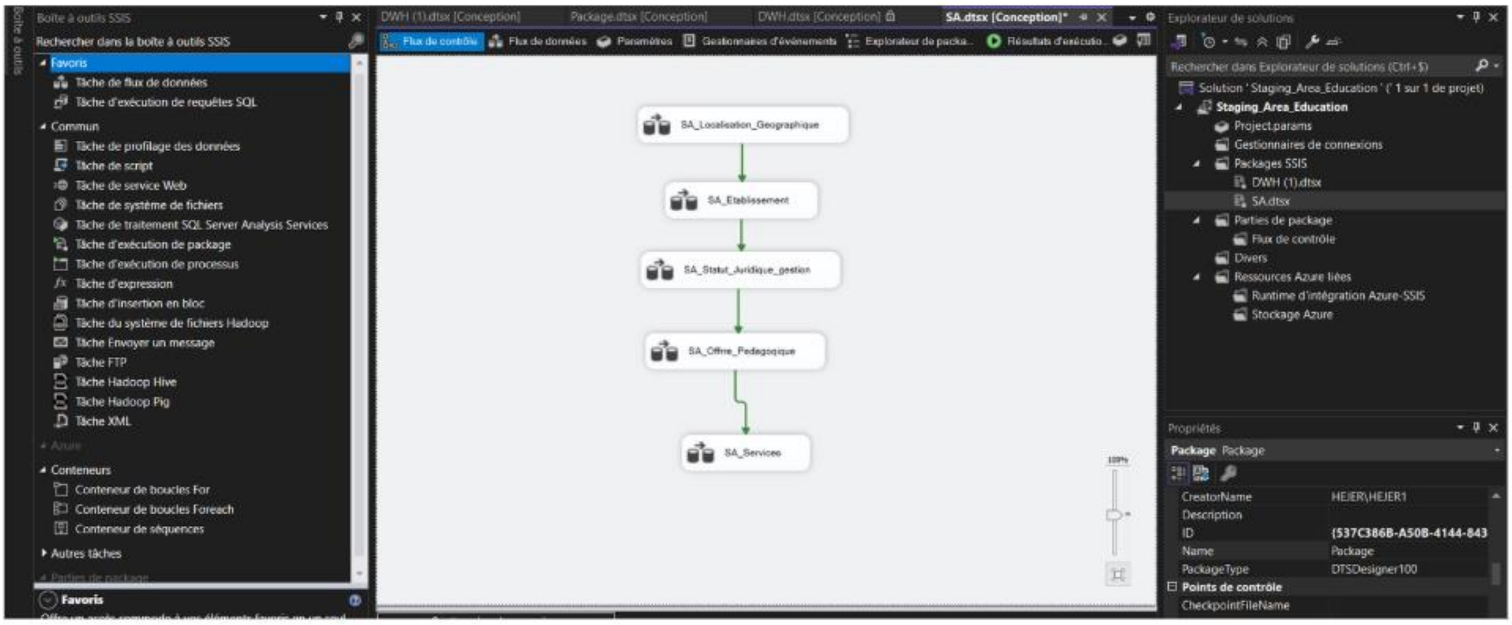
Based on the data collected, we observed a fluctuation in student numbers across different schools. The purpose of this project is to examine the causes behind the decline in enrollment and to suggest strategies that could enhance school participation.

2-Conceptual Model of the Data :



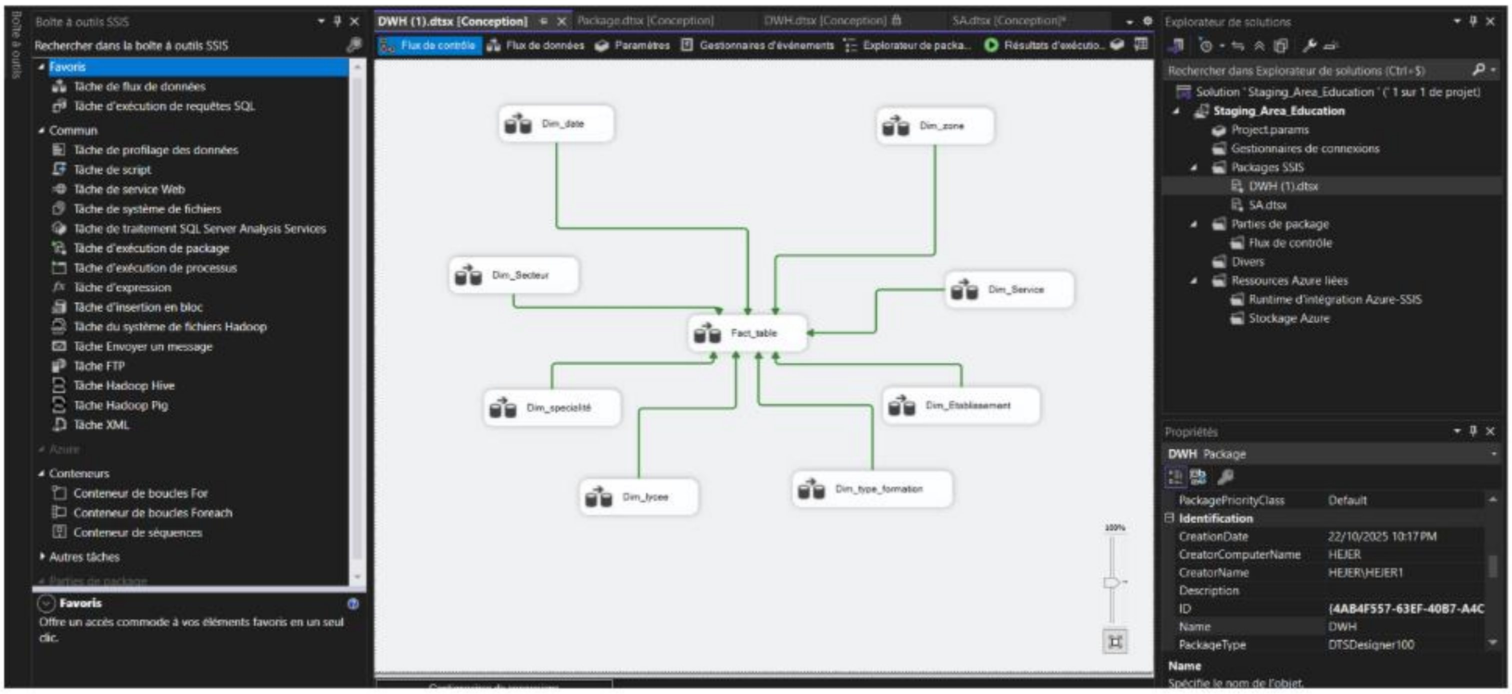
3-SSIS Packages :

Staging area :



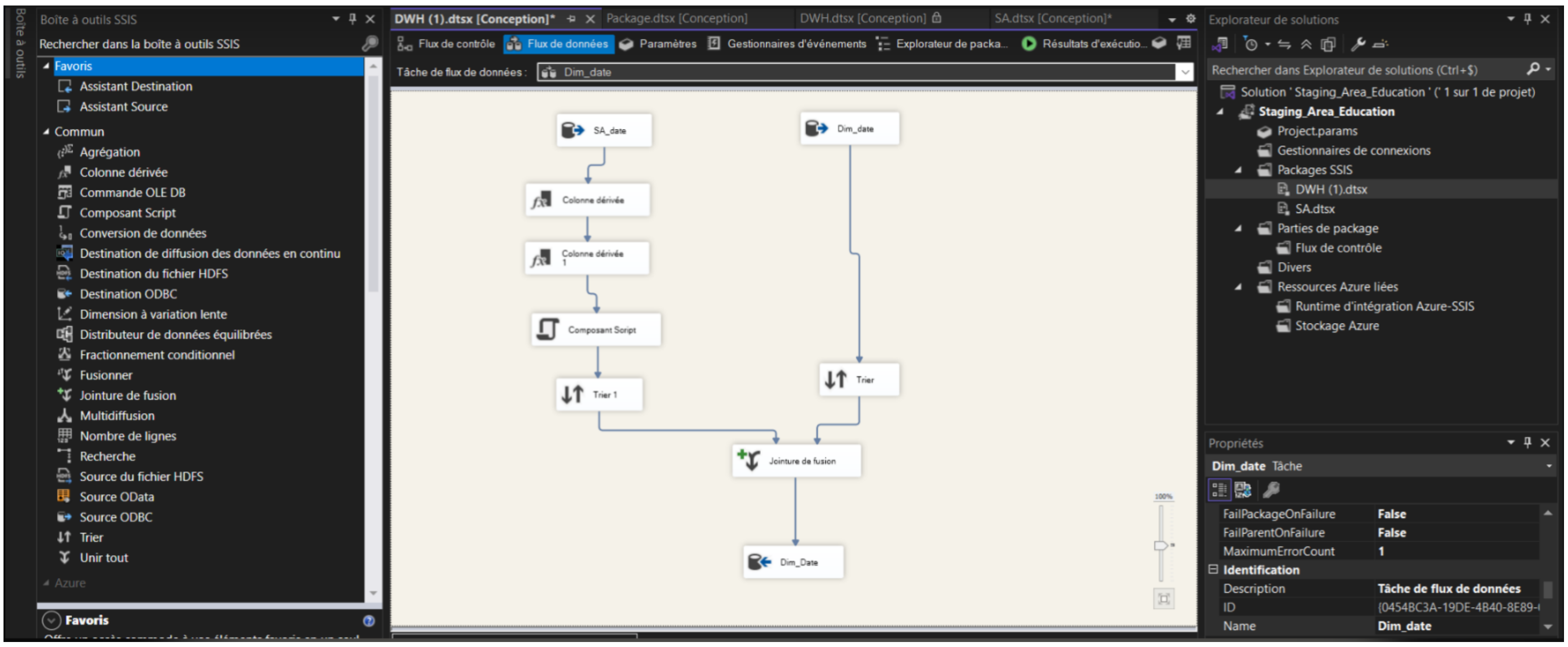
CreatorName HEJER\HEJER1
ID {537C386B-A50B-4144-843
Name Package
PackageType DTSDesigner100

Data Warehouse:



PackagePriorityClass Default
CreationDate 22/10/2025 10:17 PM
CreatorComputerName HEJER
CreatorName HEJER\HEJER1
ID {4AB4F557-63EF-40B7-A4C
Name DWH
PackageType DTSDesigner100

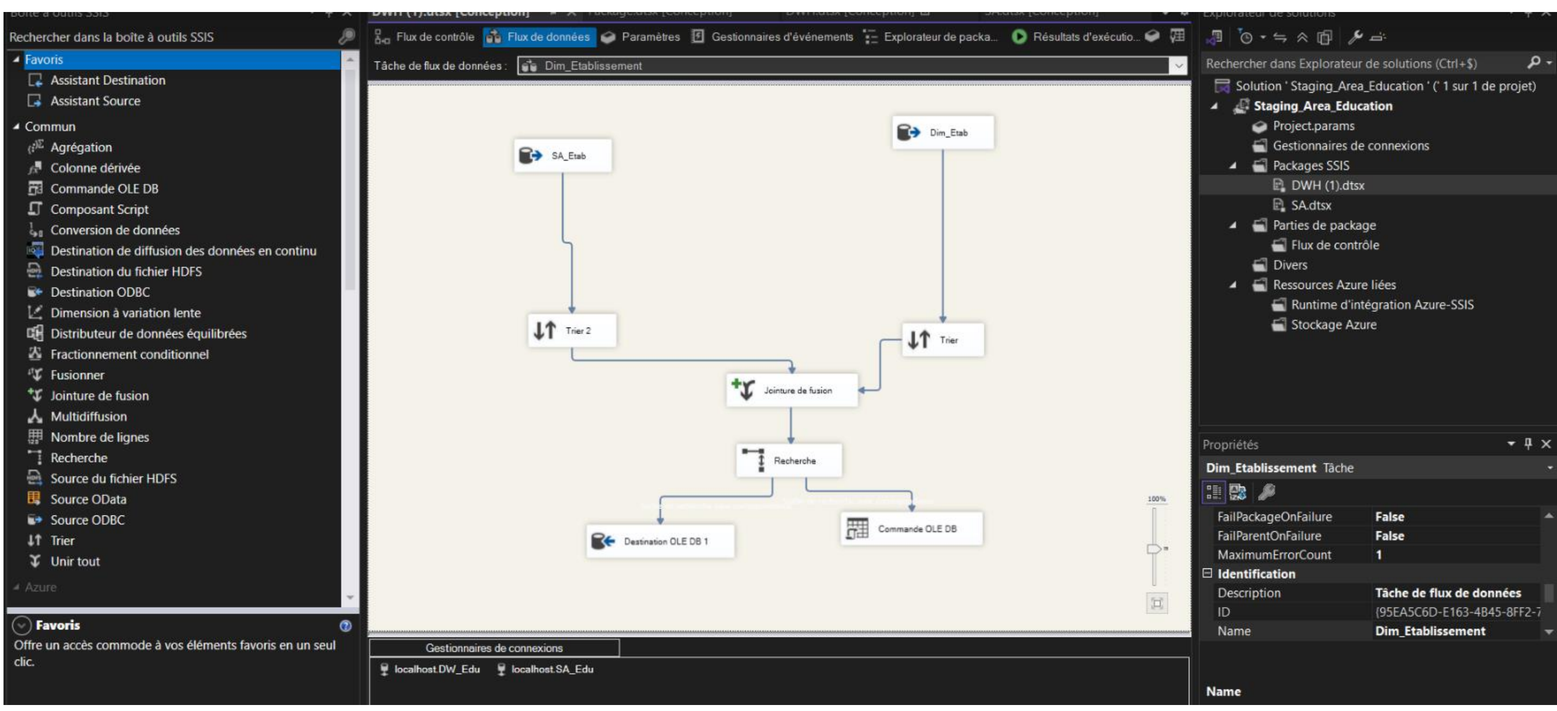
A-Dim_date:



Dim_date Tâche

FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{0454BC3A-19DE-4B40-8E89-
Name	Dim_date

B-Dim_Etablissement:

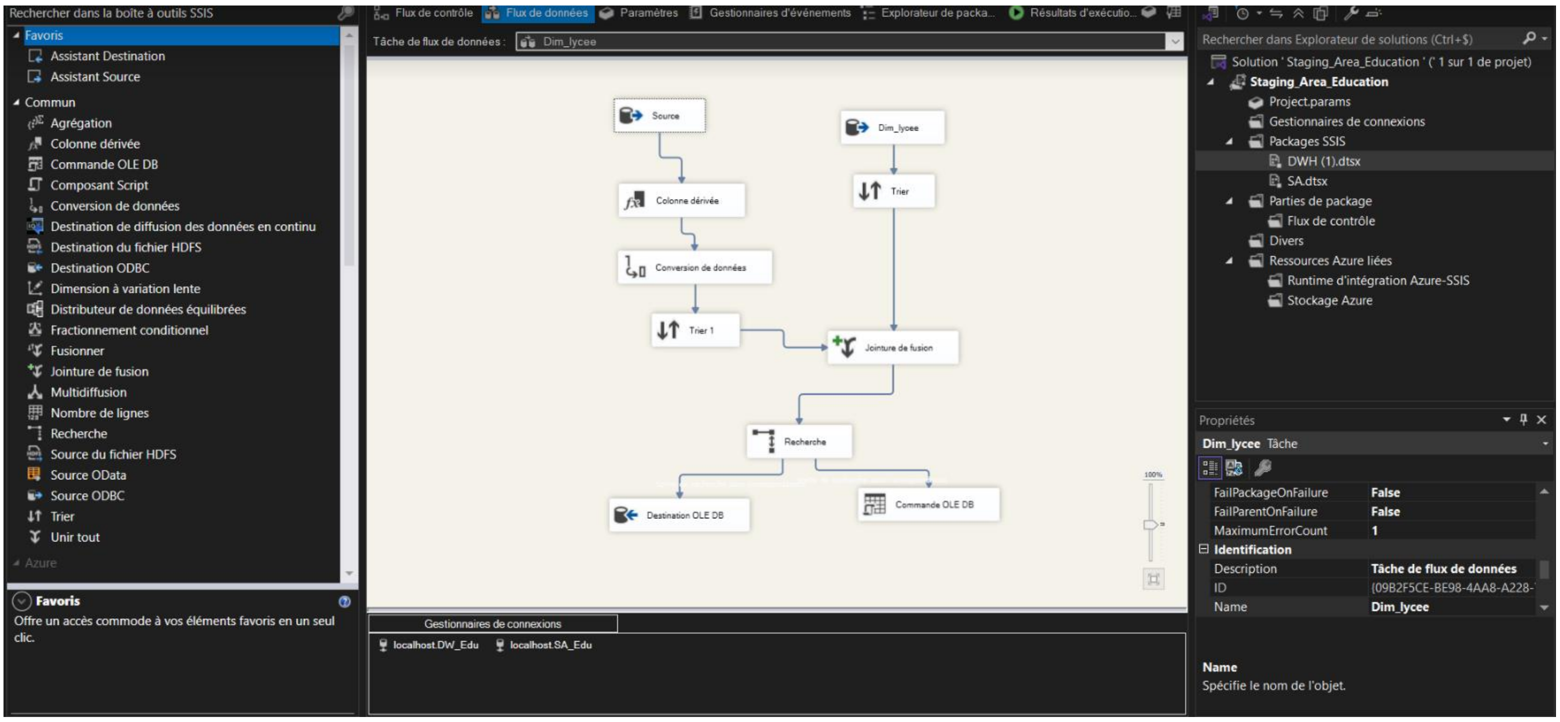


Dim_Etablissement Tâche

FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{95EA5C6D-E163-4B45-8FF2-7
Name	Dim_Etablissement

Name

C-Dim_Lycee:



Favoris
Offre un accès commode à vos éléments favoris en un seul clic.

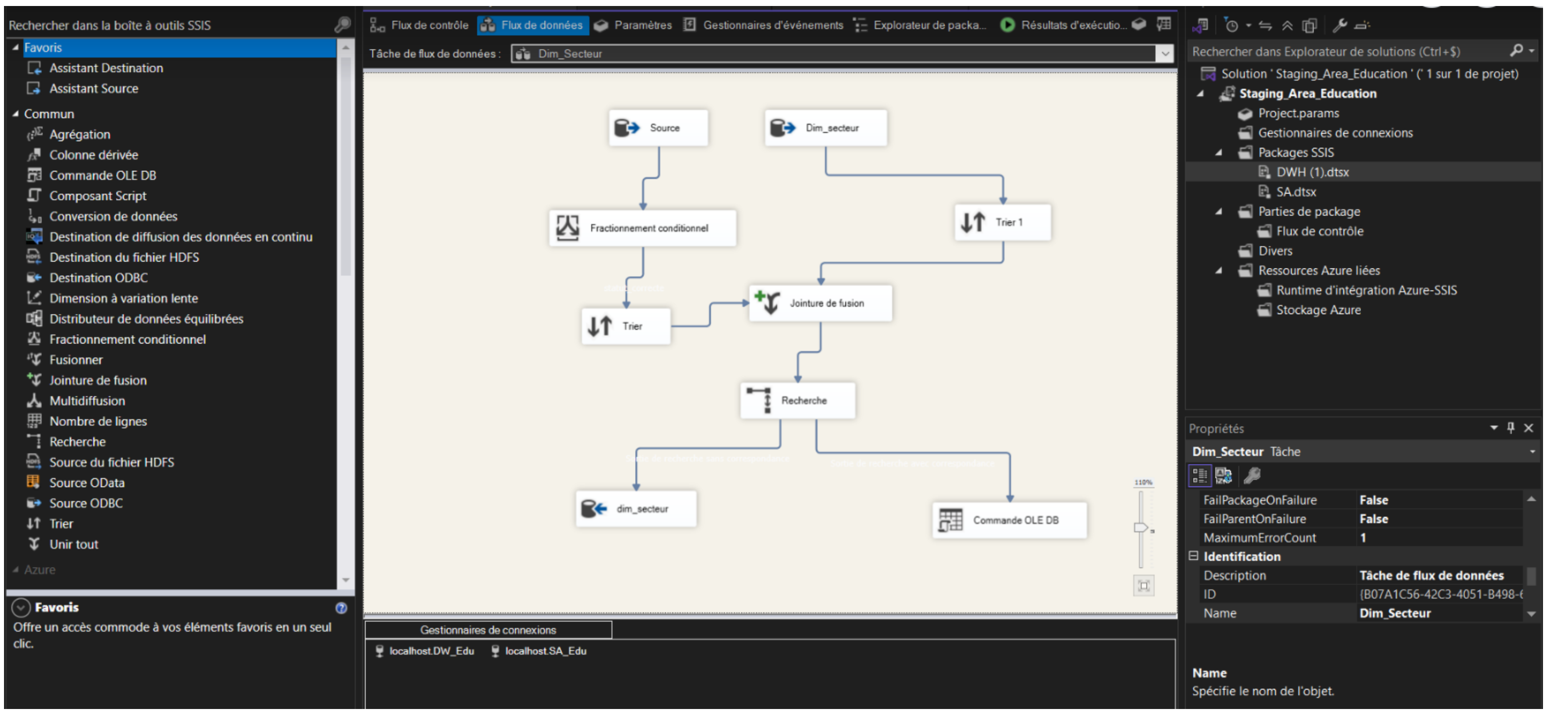
Tâche de flux de données : Dim_lycee

Dim_lycee Tâche

FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{09B2F5CE-BE98-4AA8-A228-
Name	Dim_lycee

Name
Spécifie le nom de l'objet.

D-Dim_Secteur:



Favoris
Offre un accès commode à vos éléments favoris en un seul clic.

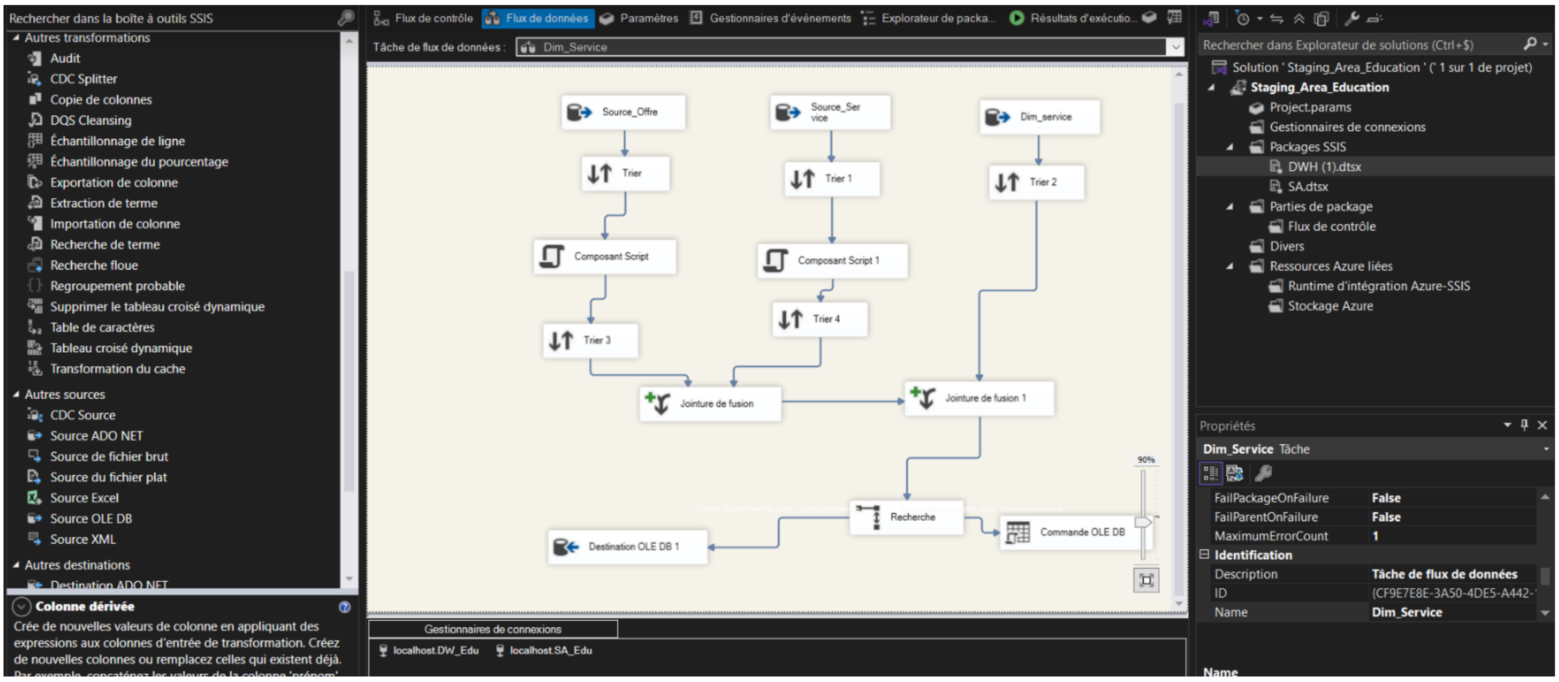
Tâche de flux de données : Dim_Secteur

Dim_Secteur Tâche

FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{B07A1C56-42C3-4051-B498-(
Name	Dim_Secteur

Name
Spécifie le nom de l'objet.

E-Dim_Service:

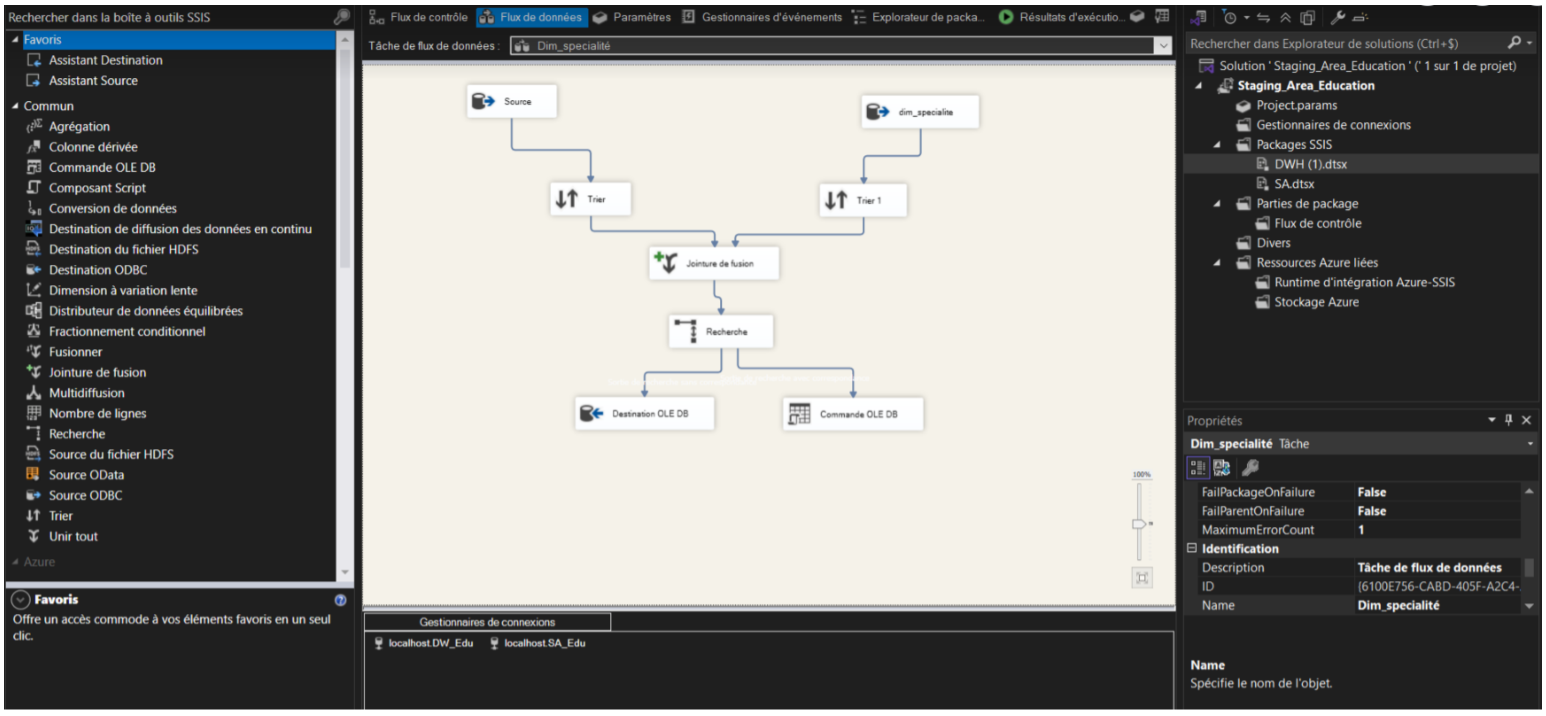


Colonne dérivée

Crée de nouvelles valeurs de colonne en appliquant des expressions aux colonnes d'entrée de transformation. Créez de nouvelles colonnes ou remplacez celles qui existent déjà. Par exemple, concaténez les valeurs de la colonne 'prénom'

FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{CF9E7E8E-3A50-4DE5-A442-
Name	Dim_Service

F-Dim_Specialité:



Favoris

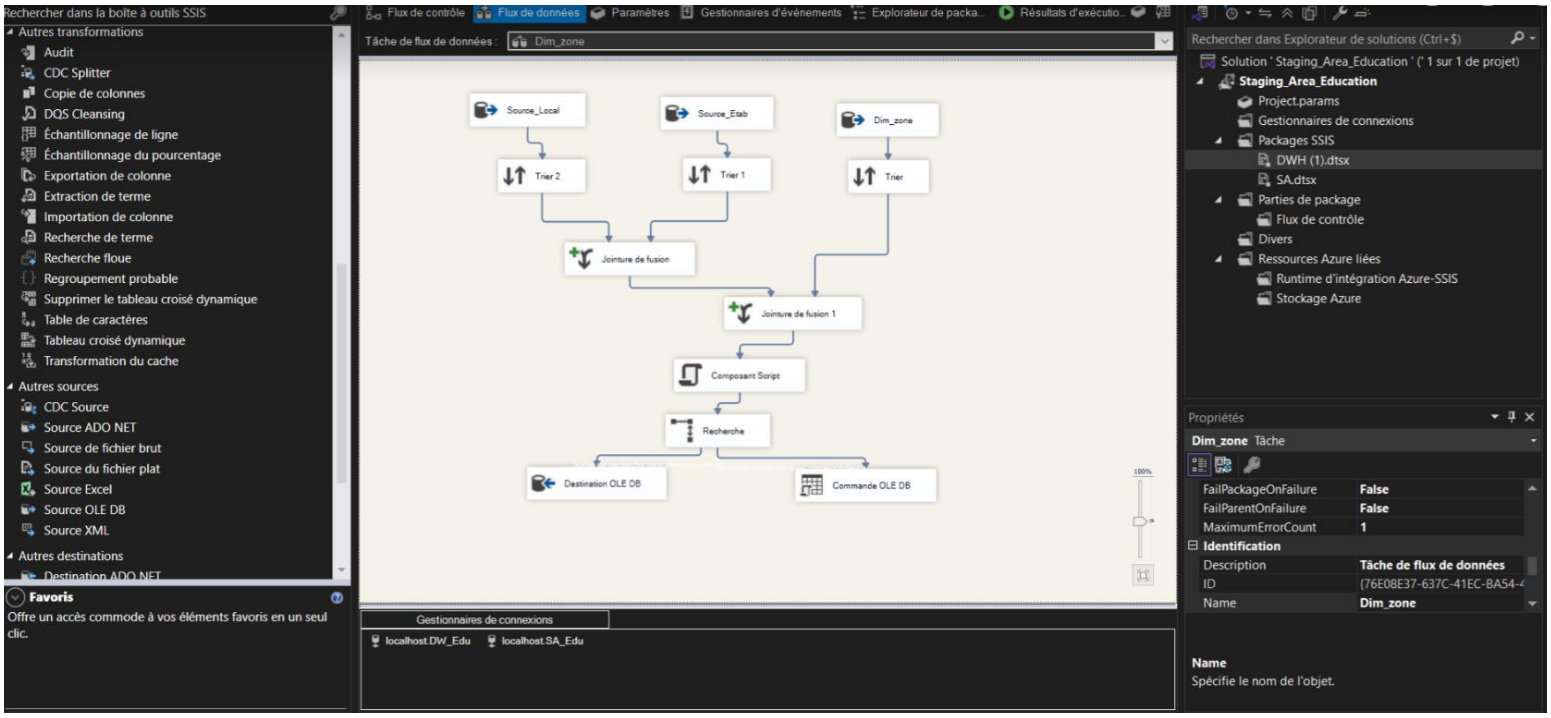
Offre un accès commode à vos éléments favoris en un seul clic.

FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{6100E756-CABD-405F-A2C4-
Name	Dim_specialité

Name

Spécifie le nom de l'objet.

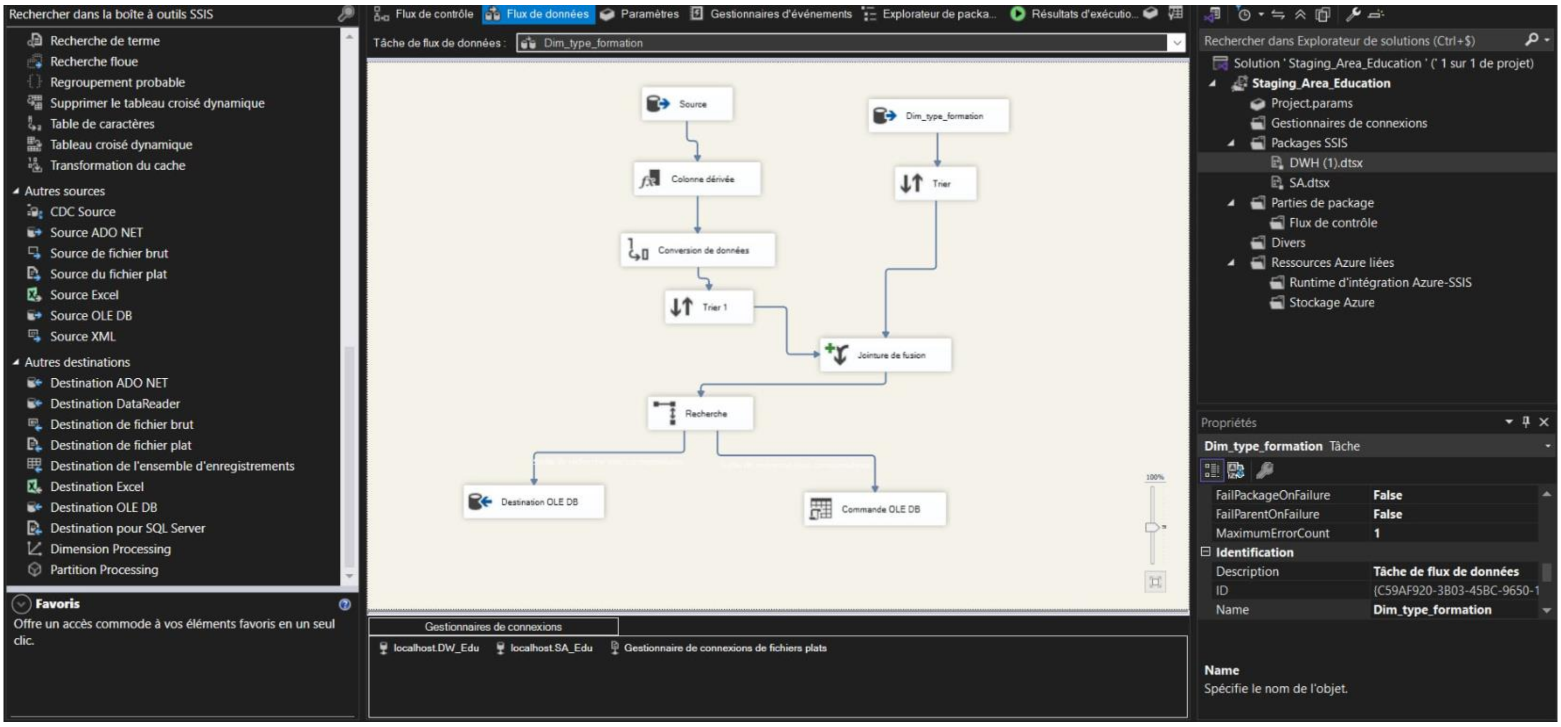
G-Dim_Zone :



FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{76E0BE37-637C-41EC-BA54-4
Name	Dim_zone

Name
Spécifie le nom de l'objet.

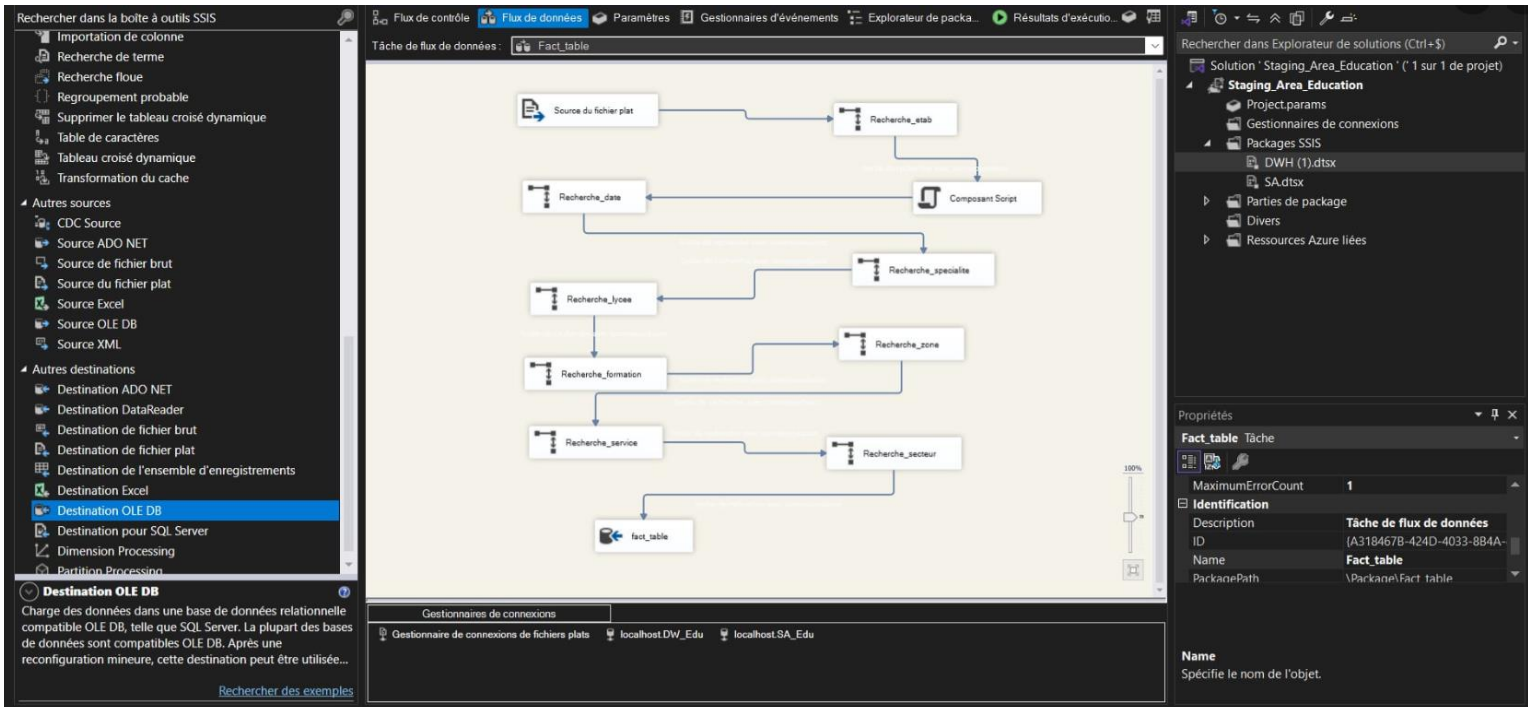
G-Dim_Type_formation :



FailPackageOnFailure	False
FailParentOnFailure	False
MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{C59AF920-3B03-45BC-9650-1
Name	Dim_type_formation

Name
Spécifie le nom de l'objet.

H-Fact_table:



Destination OLE DB

Charge des données dans une base de données relationnelle compatible OLE DB, telle que SQL Server. La plupart des bases de données sont compatibles OLE DB. Après une reconfiguration mineure, cette destination peut être utilisée...

MaximumErrorCount	1
Identification	
Description	Tâche de flux de données
ID	{A318467B-424D-4033-8B4A-
Name	Fact_table
PackagePath	\Package\Fact_table

Name
Spécifie le nom de l'objet.

4-Description of the ETL Process :

A-Dim_date:

For the Date dimension, we used a source from the staging area containing the opening dates. A derived column was applied to correct the date format, and another derived column was used to extract the day, month, year, and semester. A Script Component was then used to automatically generate the Date ID. After sorting, a left join was performed with the sorted Date dimension itself, and in the destination, only new records were added.

B-Dim_Etablissement :

For the *Establishment* dimension, we used the Dimension table from the staging area as our primary source, along with additional data

from the operational database. We first filtered the records based on the establishment ID, then applied a left merge join to align the data from both sources. After the join, a lookup was performed using the establishment ID: if the ID already exists in the target table, an update is executed using an OLE DB Command; if the ID does not exist, the record is inserted into the destination table.

C-Dim_Lycee:

For the *High School* dimension, the source in the staging area contains separate columns such as *Lycée Agricole*, *Lycée Militaire*, and *Lycée des Métiers*, each with values 0 or 1. We created a derived column called *School_Type*, which assigns the type based on which of these fields equals 1. After that, we applied the same process as for the Establishment dimension: filtering by ID, performing a left join with the database source, and using a lookup to check if the ID

already exists. If it does, we update the record using an OLE DB Command; if not, we insert it into the destination table.

D-Dim_Secteur:

For the *Sector* dimension, we followed almost the same process as for the Establishment dimension. However, in the first source, a conditional split was applied to remove accents and special characters (for example, converting “Privé” to “Prive”) to ensure compatibility with systems that do not accept special characters. After cleaning the data, we continued with the usual steps: filtering by sector ID, performing a left join with the database source, and using a lookup to update existing records or insert new ones.

E-Dim_Service:

For the *Service* dimension, we retrieved the columns from two tables in the staging area. Two sorts were applied to prevent duplicate

records during the merge process. An inner join was then performed to combine the sources, followed by a left join with the existing Service dimension. Finally, a lookup was used to determine whether to update existing records or insert new ones into the destination table.

F-Dim_Specialité:

For the *Specialty* dimension, we applied almost the same process as for the Establishment dimension. The columns were retrieved from the staging area as the main source, and the data was filtered by specialty ID. Then, a left join was performed with the database source, followed by a lookup to check whether the specialty ID already exists. If the ID exists, the record is updated using an OLE DB Command; if not, it is inserted into the destination table.

G-Dim_Zone :

For the *Zone* dimension, we used two sources from the staging area: the *Local* and *Establishment* tables, from which we extracted the required columns. A sort was applied to each source, followed by an inner join to merge the data. This result was then joined with the existing Zone dimension. A Script Component was used to generate the Zone ID, and finally, a lookup was performed to determine whether to update the existing record or insert a new one into the destination table.

G-Dim_Type_formation :

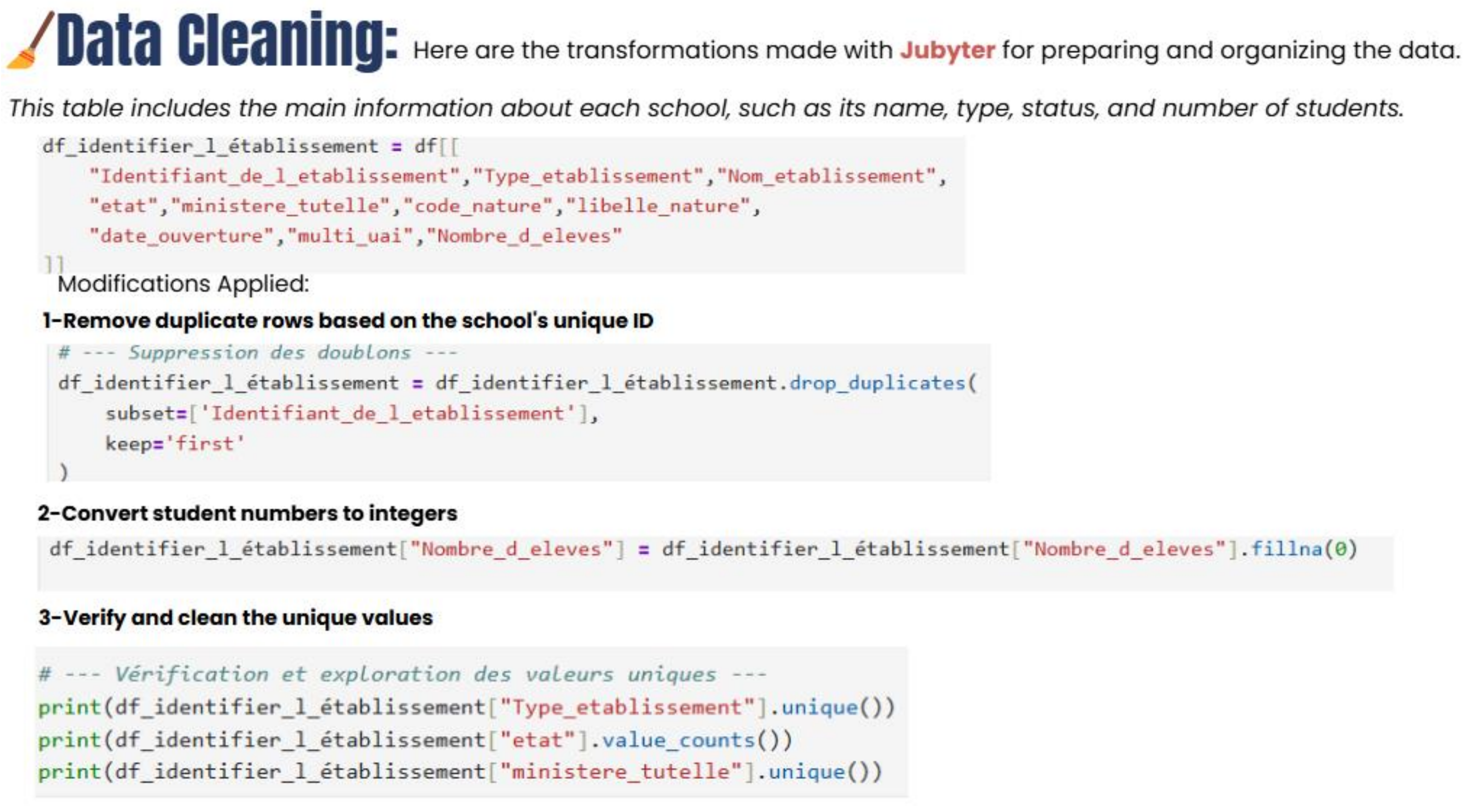
For the *Type_Formation* dimension, the process was similar to the High School dimension. The source contained separate columns for *General Path*, *Professional Path*, and *Technological Path*, each with values 0 or 1. A derived column called *Path_Type* was created to

assign the path name based on which column has a value of 1. The same process was then applied: filtering by ID, performing a left join with the database source, and using a lookup to update existing records or insert new ones into the destination table.

H-Fact_table :

For the *Fact* table, data was first extracted from a flat file in the staging area. Lookup transformations were then used to retrieve the corresponding dimension keys (such as DateID, EstablishmentID, SpecialtyID, etc.). Finally, the enriched records containing these foreign keys were inserted into the fact table.

2-Data Cleaning:



Data Cleaning: Here are the transformations made with **Jubyter** for preparing and organizing the data.

This table includes the main information about each school, such as its name, type, status, and number of students.

```
df_identifier_l_établissement = df[[
    "Identifiant_de_l_etablissement","Type_etablissement","Nom_etablissement",
    "etat","ministere_tutelle","code_nature","libelle_nature",
    "date_ouverture","multi_uai","Nombre_d_eleves"
]]
```

Modifications Applied:

1-Remove duplicate rows based on the school's unique ID

```
# --- Suppression des doublons ---
df_identifier_l_établissement = df_identifier_l_établissement.drop_duplicates(
    subset=['Identifiant_de_l_etablissement'],
    keep='first'
)
```

2-Convert student numbers to integers

```
df_identifier_l_établissement["Nombre_d_eleves"] = df_identifier_l_établissement["Nombre_d_eleves"].fillna(0)
```

3-Verify and clean the unique values

```
# --- Vérification et exploration des valeurs uniques ---
print(df_identifier_l_établissement["Type_etablissement"].unique())
print(df_identifier_l_établissement["etat"].value_counts())
print(df_identifier_l_établissement["ministere_tutelle"].unique())
```

This table shows each school is public or private and gives details about its administrative region and contract type.

```python
df_Statut_juridique_et_gestion = df[[
    "Statut_public_prive","Type_contrat_prive","Code_academie",
    "etat","Libelle_academie","Code_departement",
    "Libelle_departement","Code_region","Libelle_region"
]]
```

Modifications Applied:

1-Remove rows where the Public/Private Status (Statut_public_prive) is equal to a hyphen ("-").

```python
# --- Suppression des lignes où Statut_public_prive == "-" ---
df_Statut_juridique_et_gestion = df_Statut_juridique_et_gestion[
    df_Statut_juridique_et_gestion["Statut_public_prive"] != "-"
]
```

2-Replace missing values in the Public/Private Status with "UNKNOWN"

```python
# --- Remplacement des valeurs manquantes dans Statut_public_prive par "INCONNU" ---
df_Statut_juridique_et_gestion["Statut_public_prive"] = df_Statut_juridique_et_gestion["Statut_public_prive"].fillna("INCONNU")
```

3-Remove rows where the Private Contract Type is still missing afterwards.

```python
# --- Suppression des lignes où Type_contrat_prive est encore manquant ---
df_Statut_juridique_et_gestion = df_Statut_juridique_et_gestion.dropna(subset=['Type_contrat_prive'])
```

4-Verify for duplicates and check the unique values (in all relevant columns)

```python
# --- Vérification des doublons et des valeurs uniques ---
print(df_Statut_juridique_et_gestion["Statut_public_prive"].value_counts())
print(df_Statut_juridique_et_gestion["Type_contrat_prive"].unique())
print(df_Statut_juridique_et_gestion["Code_region"].value_counts())
```

This table describes the different types of education offered in each school, such as general, technological, or vocational paths.

```python
# --- Sélection des colonnes liées à l'offre pédagogique ---
df_Offre_pédagogique = df[[
    "Ecole_maternelle", "Ecole_elementaire", "Voie_generale",
    "Voie_technologique", "Voie_professionnelle", "Post_BAC",
    "Lycee_des_metiers", "Lycee_Agricole", "Lycee_militaire",
    "GRETA", "ULIS", "Segpa", "Apprentissage", "Section_arts",
    "Section_cinema", "Section_theatre", "Section_sport",
    "Section_internationale", "Section_europeenne",
    "rpi_concentre", "rpi_disperse"
]].copy()
```

Modifications Applied:

1-Replace all missing values (NaN) in the educational offering columns with the number 0.

```python
# --- Remplacement des valeurs manquantes par 0 ---
df_Offre_pédagogique = df_Offre_pédagogique.fillna(0)
```

2-Convert all indicator columns (except for rpi_disperse) into whole numbers (integers).

```python
# --- Conversion de toutes les colonnes sauf 'rpi_disperse' en int ---
colonnes_a_convertir = df_Offre_pédagogique.columns.difference(['rpi_disperse'])
df_Offre_pédagogique[colonnes_a_convertir] = df_Offre_pédagogique[colonnes_a_convertir].astype(int)
```

3-Verify the consistency of all indicator columns to ensure values are only 0 or 1

```python
# --- Vérification de cohérence des valeurs (0 ou 1) ---
for col in df_Offre_pédagogique.columns:
    valeurs_uniques = df_Offre_pédagogique[col].unique()
    print(f"{col} : {valeurs_uniques}")
```

This table provides information about available services like catering, housing, and priority education areas.

```
# --- Sélection des colonnes liées à la population et aux services ---
df_Population_et_services = df[[
    "Restauration", "Hebergement", "Appartenance_Education_Prioritaire"
]].copy()
```

Modifications Applied:

1-Fill missing values in Catering and Boarding with 0, then convert to whole numbers.

```
# --- Remplacement des valeurs manquantes dans Restauration et Hebergement par 0 ---
df_Population_et_services[["Restauration", "Hebergement"]] = (
    df_Population_et_services[["Restauration", "Hebergement"]].fillna(0)
)
```

2-Conversion of Catering and Boarding Columns to Integers

```
# --- Conversion des deux colonnes en entiers ---
df_Population_et_services[["Restauration", "Hebergement"]] = (
    df_Population_et_services[["Restauration", "Hebergement"]].astype(int)
)
```

3-Cleaning and Standardizing the Priority Education Status Column

```
# --- Nettoyage de la colonne Appartenance_Education_Prioritaire ---
df_Population_et_services["Appartenance_Education_Prioritaire"] = (
    df_Population_et_services["Appartenance_Education_Prioritaire"]
        .fillna("AUCUNE")   # Remplacer NaN par "AUCUNE"
        .str.strip()        # Supprimer les espaces avant/après
        .str.upper()        # Mettre en majuscules
)
```

This table includes the school's location data such as postal code, city name, latitude, and longitude.

```
# --- Sélection des colonnes liées à la localisation géographique ---
df_Localisation_géographique = df[[
    "Code_postal", "Code_commune", "Nom_commune",
    "latitude", "longitude", "position"
]]
```

Modifications Applied:

1-Removing Rows with Missing Geographical Data

```
# --- Suppression des lignes avec valeurs manquantes dans latitude, longitude ou position ---
df_Localisation_géographique = df_Localisation_géographique.dropna(subset=['latitude'])
df_Localisation_géographique = df_Localisation_géographique.dropna(subset=['longitude'])
df_Localisation_géographique = df_Localisation_géographique.dropna(subset=['position'])
```

2-Verification of Geographical Data Cleaning

```
# --- Vérification du nettoyage ---
df_Localisation_géographique.isnull().sum()
df_Localisation_géographique.head()
```

3-Normalization of Names (Accent Removal)

```
df_final['Nom_etablissement'] = [
    unidecode(x) if isinstance(x, str) else x
    for x in df_final['Nom_etablissement']
]
```