Data science project

Exploration et analyse de données

Fatma mrabti
Dorra Bouzidi
Oussema khalifa ben mimouna
Mohamed ali lamouchi
Fatma mhadhbi
Aymen kefi

Sommaire

1-Objectif

L'objectif principal du code est de combiner les données dispersées dans plusieurs fichiers CSV en une seule entité structurée, représentée sous la forme d'un DataFrame pandas.

2- Technologies utilisées

Environnement de developpement

Bibliothéque

Langage de programmation

Python

3- Extraction et nettoyage des données

3.1 Extraction des donnés

Dans cette partie on a extrait les données qu'on a selectionner selon les categories

Script

```
import pandas as pd
import glob

#usagers
# Get a list of all CSV files in the directory
files = glob.glob('DataScienceProject/Data/usagers*.csv')

# Initialize an empty DataFrame to store the combined data
combined_data = pd.DataFrame()
print(files)
# Loop through each file and append its data to the combined DataFrame
for file in files:
    # Read the first row of the file to infer data types
    dtypes = pd.read_csv(file, nrows=1).dtypes.to_dict()

    # Read the entire CSV file using inferred data types
    df = pd.read_csv(file, dtype=dtypes, encoding='UTF-8', sep=';', quotechar='"')
    combined_data = pd.concat([combined_data, df], ignore_index=True)

# Now 'combined_data' contains data from all CSV files
combined_data.to_excel('DataScienceProject/combinedData/usagers.xlsx', index=False)
combined_data.head
```

Details du script

1- Identification des fichiers

- Utilise le module glob pour obtenir une liste de chemins de fichiers pour les fichiers CSV correspondant à un motif spécifié.

2- Combinaison des données

- Utilise une compréhension de liste pour lire et concaténer les données de chaque fichier CSV dans un seul DataFrame pandas (combined_data). Spécifie l'encodage UTF-8, le point-virgule (comme séparateur et les guillemets doubles (") comme caractère de citation lors de la lecture des fichiers.

3- Sauvegarde des données

- Enregistre les données combinées dans un fichier Excel nommé 'usagers.xlsx' dans un répertoire désigné ('DataScienceProject/combinedData/'). Le paramètre index=False garantit que l'index du DataFrame est exclu du fichier Excel.

3- Résultat

3.1 Résultat

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Num_Acc	id_usager	d_vehicule	num_veh	place	catu	grav	sexe	an_nais	trajet	secu1	secu2	secu3	locp	actp	etatp
2	2,02E+11	267 638	201 764	B01	1	1	3	1	2000	1	0	9	-1	0	0	-1
3	2,02E+11	267 639	201 765	A01	1	1	1	1	1978	1	1	-1	-1	0	0	-1
4	2,02E+11	267 636	201 762	A01	1	1	4	1	1983	0	1	-1	-1	0	0	-1
5	2,02E+11	267 637	201 763	B01	1	1	3	1	1993	0	1	-1	-1	0	0	-1
6	2,02E+11	267 634	201 761	A01	1	1	1	1	1995	1	1	0	-1	0	0	-1
7	2,02E+11	267 635	201 761	A01	10	3	3	2	1959	4	0	-1	-1	3	3	1
8	2,02E+11	267 631	201 758	A01	1	1	1	1	2000	-1	-1	0	-1	-1	-1	-1
9	2,02E+11	267 632	201 759	D01	1	1	2	2	2014	5	0	-1	-1	-1	-1	-1
10	2,02E+11	267 627	201 754	A01	1	1	4	2	1997	1	1	-1	-1	-1	-1	-1
11	2,02E+11	267 628	201 755	Z01	1	1	-1	-1		-1	8	-1	-1	-1	-1	-1
12	2,02E+11	267 625	201 752	B01	1	1	4	1	2009	2	0	-1	-1	0	0	-1
13	2,02E+11	267 626	201 753	A01	1	1	1	1	1976	5	1	-1	-1	0	0	-1
14	2,02E+11	267 622	201 750	B01	4	2	4	2	2002	0	1	0	-1	-1	-1	-1
15	2,02E+11	267 623	201 750	B01	1	1	1	1	2001	5	1	5	-1	-1	-1	-1
16	2,02E+11	267 624	201 751	A01	1	1	4	1	1991	0	1	5	-1	-1	-1	-1
17	2,02E+11	267 619	201 748	B01	1	1	3	1	1972	5	2	-1	-1	0	0	-1
18	2,02E+11	267 620	201 748	B01	2	2	4	2	1984	5	8	-1	-1	0	0	-1
19	2,02E+11	267 621	201 749	A01	1	1	1	1	1971	5	1	-1	-1	0	0	-1
20	2,02E+11	267 616	201 746	A01	1	1	1	1	1981	1	1	0	-1	-1	-1	-1
21	2,02E+11	267 617	201 747	B01	2	2	4	2	1936	5	1	0	-1	-1	-1	-1
22	2,02E+11	267 618	201 747	B01	1	1	1	1	1935	5	1	0	-1	-1	-1	-1
23	2,02E+11	267 611	201 744	A01	1	1	1	1	1976	5	8	8	-1	-1	-1	-1
24	2,02E+11	267 612	201 745	B01	4	2	4	2	1982	0	8	8	-1	-1	-1	-1
25	2,02E+11	267 613	201 745	B01	8	2	4	2	1960	5	8	8	-1	-1	-1	-1
26	2,02E+11	267 614	201 745	B01	8	2	4	2	1959	5	8	8	-1	-1	-1	-1
27	2,02E+11	267 615	201 745	B01	1	1	4	2	1994	4	1	8	-1	-1	-1	-1
28	2,02E+11	267 609	201 742	A01	1	1	1	1	1951	5	1	8	-1	-1	-1	-1
29	2,02E+11	267 610	201 743	B01	1	1	4	1	1989	0	8	0	-1	-1	-1	-1

3-Traitement de données

3.1 Objectif

- L'objectif principal du traitement des données avec SQL Server Integration Services (SSIS) est de faciliter l'intégration et la transformation efficaces des données.

3.1 Outil

Script

3.1 Conception

1- Composants

En utilisant des composants tels que les sources de données, les transformations et les destinations, SSIS simplifie l'extraction, la transformation selon les besoins, et le chargement des données vers un entreposage centralisé :

- **Source de fichier plat (Flat File Source)** : Ce composant permet de lire des données à partir de fichiers plats comme des fichiers CSV ou des fichiers à largeur fixe. Il est couramment employé dans le processus ETL pour extraire des données depuis des fichiers plats et les intégrer dans une base de données ou un entrepôt de données.

- **Conversion de données (Data Conversion)** : Ce composant est utilisé pour modifier le type de données d'un champ vers un autre. Par exemple, il peut convertir une chaîne de caractères en entier ou une date dans un format différent. Cela est utile lorsqu'il y a des incompatibilités de types de données entre la source et la destination.
- **Recherche (Lookup)** : Le composant de recherche est utilisé pour trouver des valeurs dans une autre table ou source de données. Il est souvent utilisé pour enrichir les données provenant d'une source avec des informations supplémentaires provenant d'une autre source, en utilisant une clé de recherche commune.
- **Destination OLE DB (OLE DB Destination)** : Ce composant permet de charger des données dans une destination OLE DB, souvent une base de données SQL Server ou une autre base de données compatible OLE DB. Il permet de spécifier la table de destination et les correspondances entre les colonnes source et destination.
- **Commande OLE DB (OLE DB Command)** : Le composant de commande OLE DB permet d'exécuter des commandes SQL spécifiques pour chaque ligne de données. Il est utile lorsque vous devez manipuler les données ou effectuer des opérations plus complexes ligne par ligne, comme la mise à jour de valeurs ou l'exécution de procédures stockées.
- **Fractionnement conditionnel fonctionnel (Functional Conditional Split)** : C'est un composant permettant de séparer les données en fonction de conditions définies par l'utilisateur, offrant une flexibilité accrue grâce à l'utilisation d'expressions complexes et de fonctions personnalisées.

4. Guide des étapes

Etape 1 : Nettoyage de données

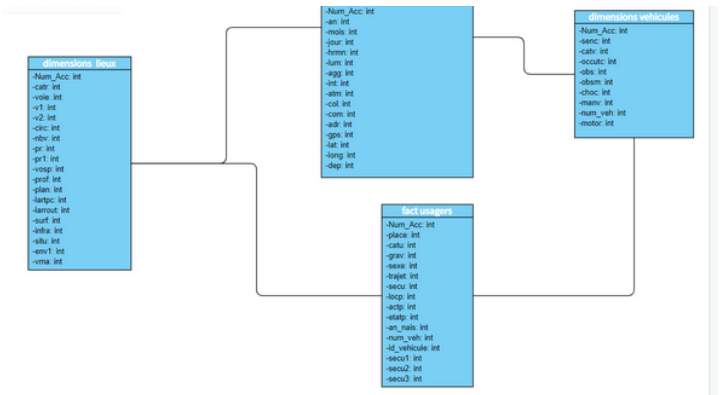
cette etape consiste a :

- Combiner les fichiers de 2017 a 2022
- Eliminier les colomnes non compatibles
- Suppression des columns inutiles
- Remplacer les valeurs vides par les valeurs les plus frequentes

```
Entrée [86]: #caracteristique
# Get a list of all CSV files in the directory

files = glob.glob('../Downloads/car*.csv')

# Initialize an empty DataFrame to store the combined data
combined_data = pd.DataFrame()
# Loop through each file and append its data to the combined DataFrame
for file in files:
    print(file)
    # Read the first row of the file to infer data types
    separator = detect_csv_separator(file)
    #dtypes = pd.read_csv(file, nrows=1,encoding='ansi').dtypes.to_dict()
    print(separator)
    # Read the entire CSV file using inferred data types
    df = pd.read_csv(file, encoding='ansi', sep=separator, quotechar='"')

    combined_data = pd.concat([combined_data, df], ignore_index=True)

# Now 'combined_data' contains data from all CSV files
combined_data['Num_Acc'] = combined_data['Num_Acc'].fillna(combined_data['Accident_Id'])
combined_data = combined_data.drop(columns=['Accident_Id'])

# Supprimer les colonnes "gps" et "dep" et "com"
combined_data.drop(columns=['gps', 'dep', 'com'], inplace=True)
column_types = df.dtypes
print(column_types)
float_columns = combined_data.select_dtypes(include=['float']).columns
print('float_columns '+float_columns)
combined_data[float_columns] = combined_data[float_columns].fillna(0)
combined_data[float_columns] = combined_data[float_columns].astype(np.int64)

string_columns = combined_data.select_dtypes(include=['object']).columns
print('string_columns '+string_columns)

adr_freq = str(valeur_plus_frequente_STR(combined_data['adr']))
print('valeur_plus_frequente adr '+ adr_freq)
combined_data['adr'].fillna(value=adr_freq, inplace=True)

lat_freq = str(valeur_plus_frequente_STR(combined_data['lat']))
print('valeur_plus_frequente lat '+ lat_freq)
combined_data['lat'].fillna(value=lat_freq, inplace=True)

long_freq = str(valeur_plus_frequente_STR(combined_data['long']))
print('valeur_plus_frequente long '+ long_freq)
```

Etape 2

Pendant cette phase, on va élaborez des tables de destination pour les dimensions et des fait afin de charger les données du datawarehouse à partir des fichiers sources Nous allons créer 3 tables de dimension et 1 table de fait pour charger les données dans le datawarehouse provenant des fichiers flat file sources



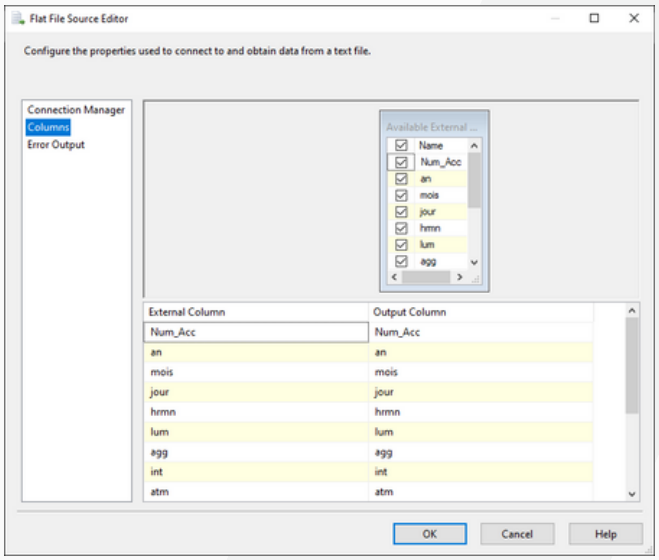
SQLQuery10.sql -...DFCG\LENOVO (62)) | DESKTOP-21CDFCG.T...o.Caracteristiques

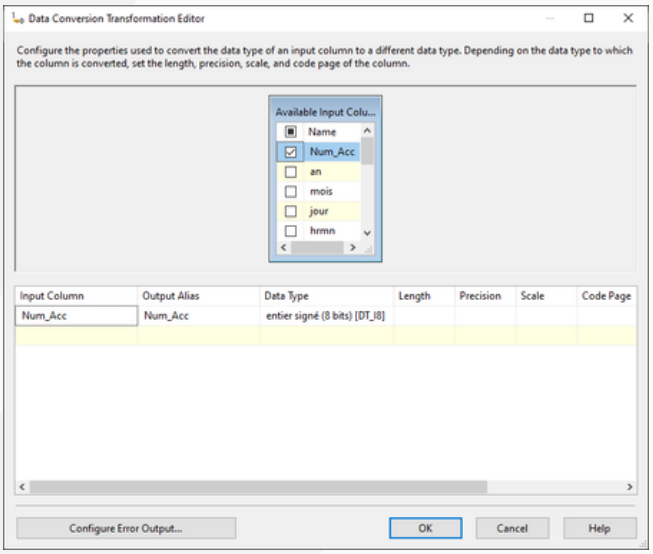
Column Name	Data Type	Allow Nulls
Caracteristiques_PK	int	☐
Num_Acc	bigint	☐
an	int	☑
mois	int	☑
jour	int	☑
hrmn	varchar(5)	☑
lum	int	☑
agg	int	☑
int	int	☑
atm	int	☑
col	int	☑
adr	varchar(255)	☑
lat	varchar(50)	☑
long	varchar(50)	☑
		☐

Etape 3

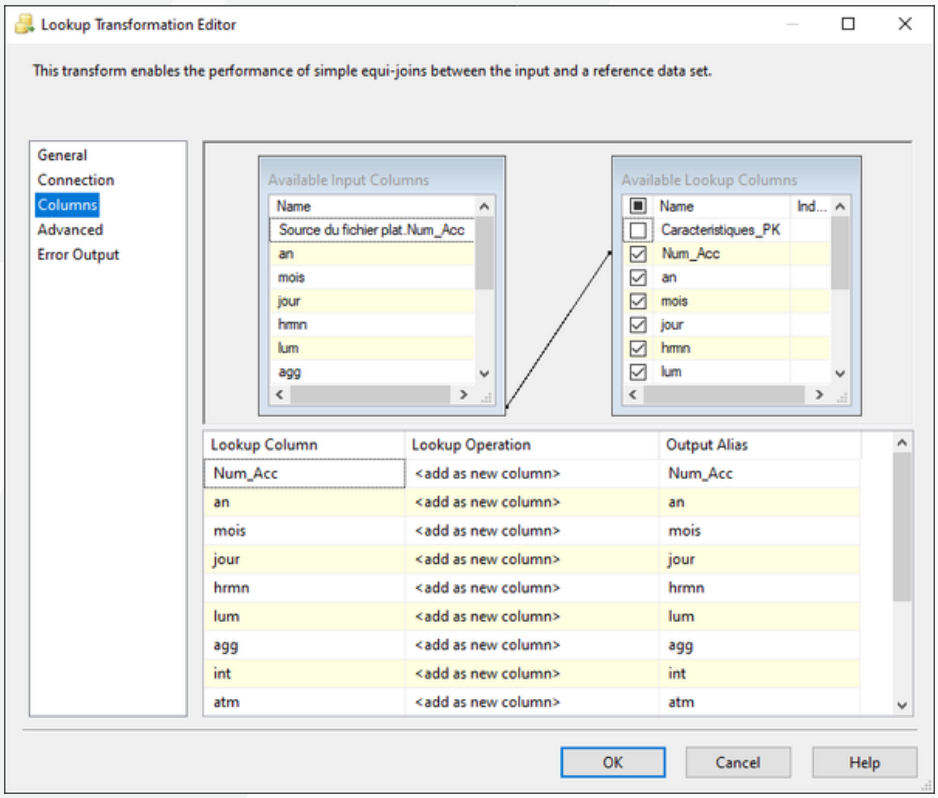
En déplaçant le flux de données (data flow) vers la tâche de flux de contrôle (control flow task), on accède ensuite au flux de données et on fait glisser le composant source (flat file source) contenant nos données extraites et enregistrées dans un fichier texte.

En utilisant le composant **"Data Conversion"**, on ajuste les types de données pour qu'ils correspondent à nos données et aux colonnes créées dans SQL Server.

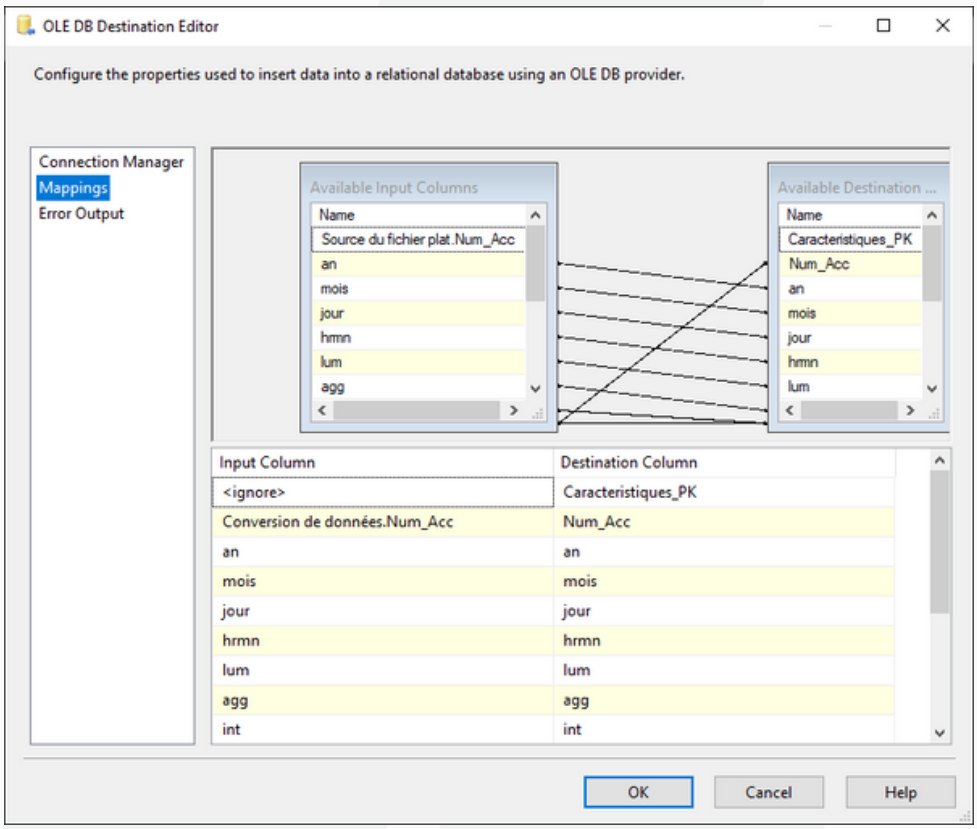




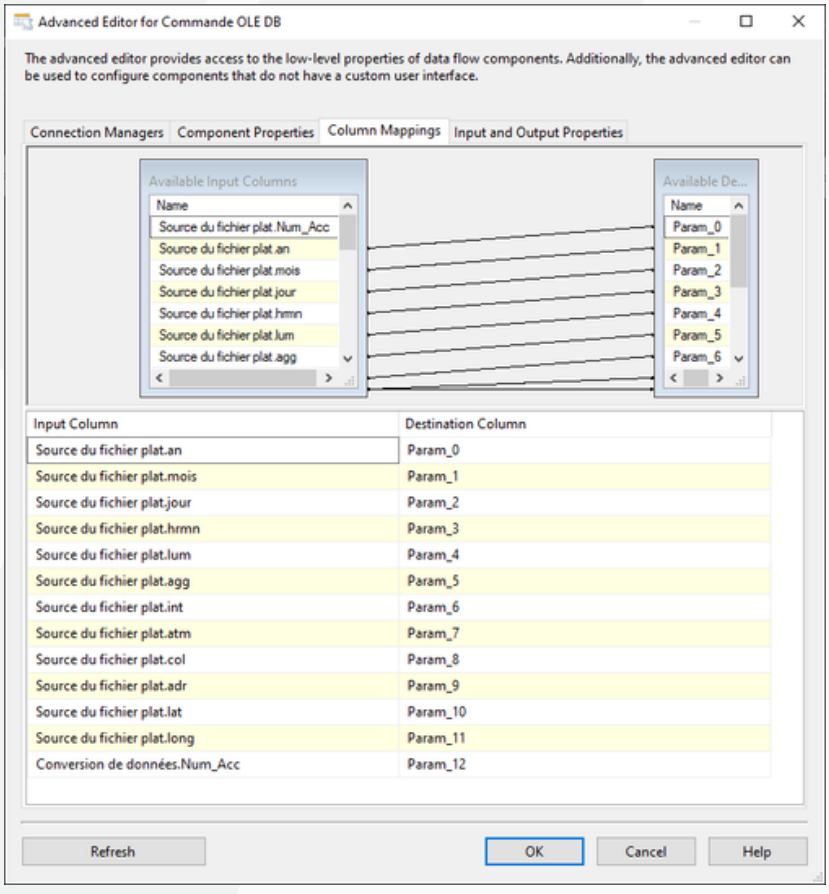
Utiliser le composant **“Lookup”** pour identifier les colonnes existantes dans le flat file et les colonnes de la table crées dans sql server



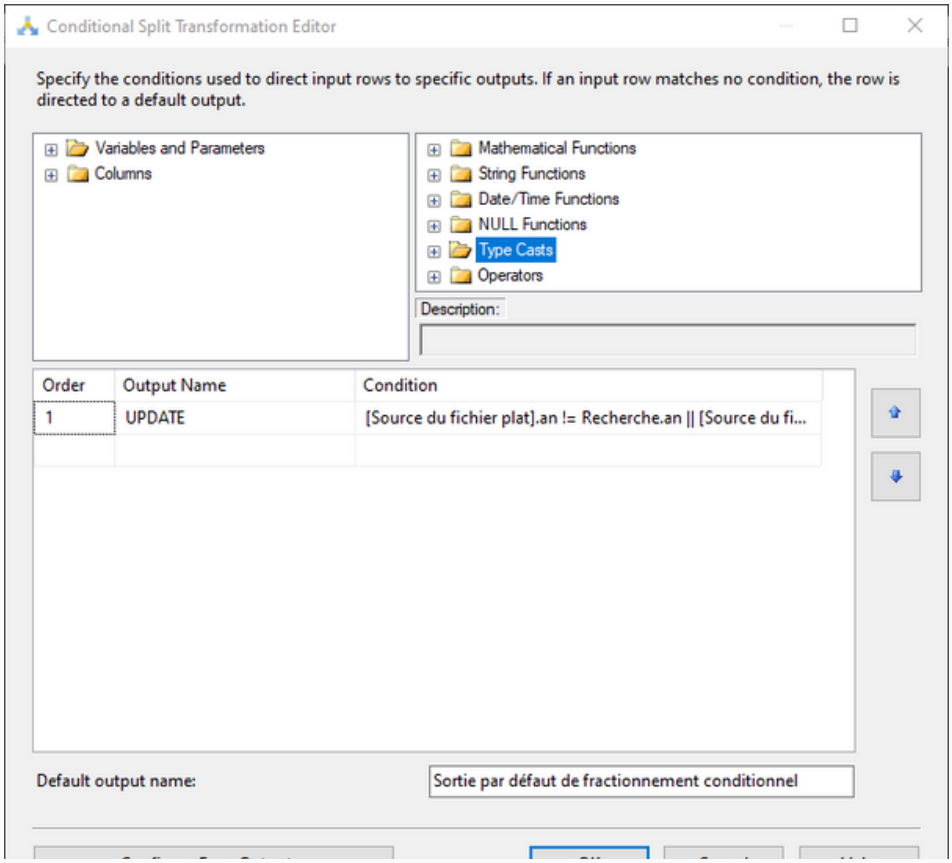
Intégrer le composant **"OLE DB Destination"** pour procéder au chargement de nos données sur SQL Server. Il est essentiel de définir clairement la table de destination et de configurer les correspondances entre les colonnes sources et celles de la destination.



Ajouter le composant **“OLE DB Command”** pour executer la comande sql de mise à jour des valeurs si existantes

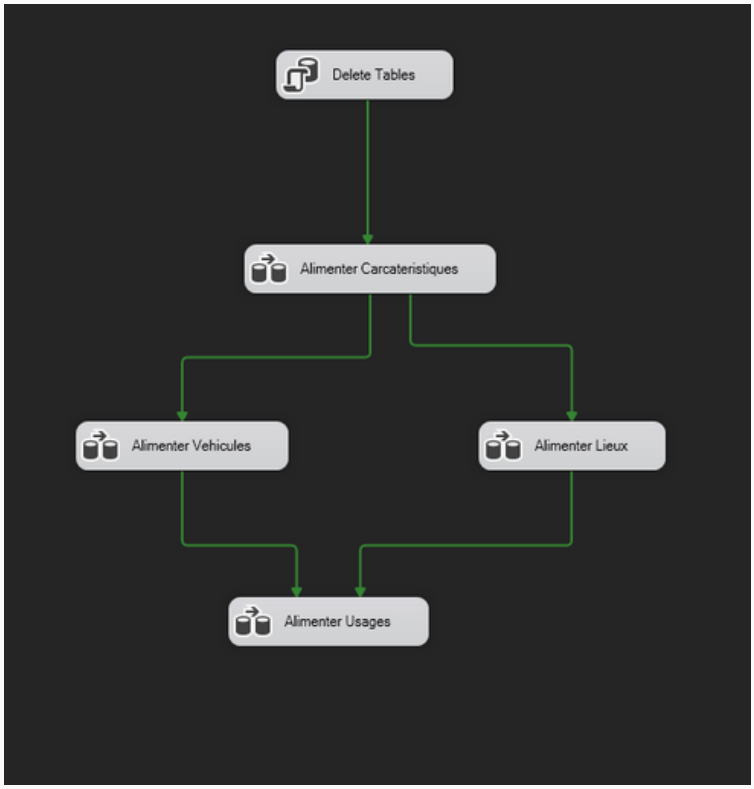


Utiliser “**Conditional Split”** pour router les données en fonction de la condition D’update afin de diviser le flux de données en plusieurs flux



Etape 3

Refaire l'étape 2 pour créer la table de fait qui contient les clés primaires



Conclusion

nous avons employé le composant Data Conversion pour ajuster les types de données, le composant Lookup pour rechercher des valeurs dans d'autres sources, le composant OLE DB Destination pour charger des données dans une destination OLE DB, et le composant OLE DB Command pour exécuter des commandes SQL sur les données avec le Conditional Split pour diviser le flux. Ces composants ont grandement facilité le processus d'extraction