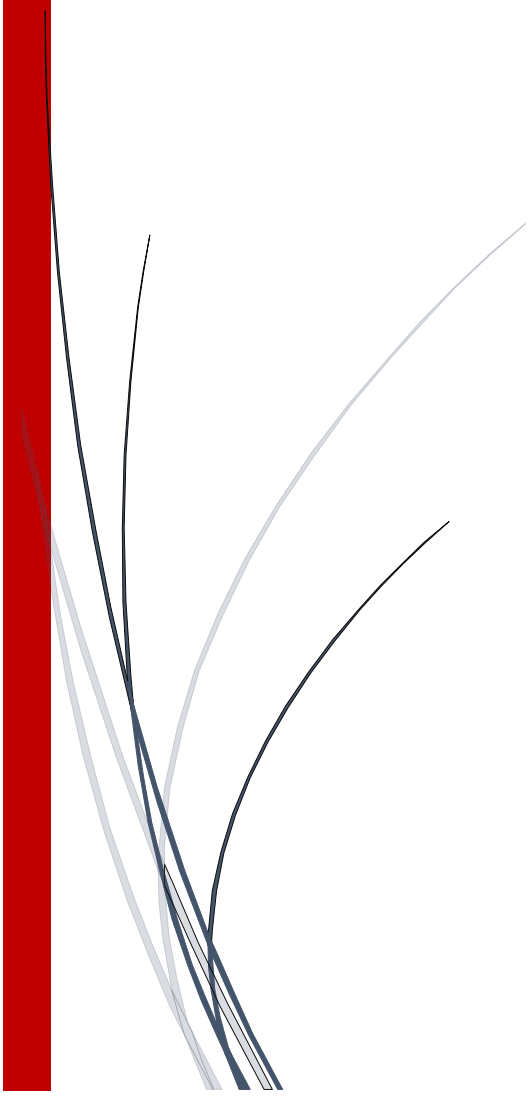
Data science Project – Collisions de véhicules

Par :

- Maissa Elamri
- Sinda Sghaier
- Safa Ben Hadj Massoud
- Nour Ben hadj yahia
- Malek bourguiba
- Aziz Bourguiba



esprit
Se former autrement

Chapitre 1 : Web scrapping

I. Objectif

L'objectif de ce script est de collecter des données sur les collisions de véhicules à New York à partir du site .
Le script utilise Selenium et BeautifulSoup pour automatiser la navigation sur le site, Pandas pour la manipulation des données tabulaires, et une pause intégrée pour permettre le chargement des pages.

II. Description des Données

Le tableau des accidents liés aux collisions de véhicules automobiles contient des détails sur l'événement d'accidents. Chaque ligne représente un événement de crash. Les tableaux de données sur les collisions de véhicules à moteur contiennent des informations sur toutes les collisions de véhicules à moteur signalées par la police à New York entre 2020 et 2021.

III. Description des variables :

Dans cette partie, on interprète les variables extraites en analysant leurs valeurs en tirant des conclusions pertinentes pour mieux comprendre les informations contenues dans le jeu de données. Cette interprétation des variables permet d'obtenir des observations significatifs et de prendre des décisions éclairées basées sur les données extraites.

CRASH DATE (Date de Collision) : Occurrence de la date à laquelle la collision s'est produite.
CRASH TIME (Heure de Collision) : Occurrence de l'heure à laquelle la collision s'est produite.
BOROUGH (Arrondissement) : Lieu où l'accident a eu lieu.
ZIP CODE (Code Postal) : Code postal où l'accident a eu lieu.
Latitude (Latitude) : Coordonnée de latitude pour le système de coordonnées global, WGS 1984, en degrés décimaux (EPSG 4326).
Longitude (Longitude) : Coordonnée de longitude pour le système de coordonnées global, WGS 1984, en degrés décimaux (EPSG 4326).
LOCATION (Emplacement) : Paire de latitude, longitude.
ON STREET NAME (Nom de la Rue) : Rue sur laquelle la collision s'est produite.
CROSS STREET NAME (Nom de la Rue Transversale) : Rue transversale la plus proche de la collision.
OFF STREET NAME (Nom de la Rue si Connu) : Adresse de la rue si elle est connue.
NUMBER OF PERSONS INJURED (Nombre de Personnes Blessées) : Nombre de personnes blessées dans la collision.

NUMBER OF PERSONS KILLED (Nombre de Personnes Tuées) : Nombre de personnes tuées dans la collision.
NUMBER OF PEDESTRIANS INJURED (Nombre de Piétons Blessés) : Nombre de piétons blessés dans la collision.
NUMBER OF PEDESTRIANS KILLED (Nombre de Piétons Tués) : Nombre de piétons tués dans la collision.
NUMBER OF CYCLIST INJURED (Nombre de Cyclistes Blessés) : Nombre de cyclistes blessés dans la collision.
NUMBER OF CYCLIST KILLED (Nombre de Cyclistes Tués) : Nombre de cyclistes tués dans la collision.
NUMBER OF MOTORIST INJURED (Nombre de Conducteurs Blessés) : Nombre de conducteurs blessés dans la collision.
NUMBER OF MOTORIST KILLED (Nombre de Conducteurs Tués) : Nombre de conducteurs tués dans la collision.
CONTRIBUTING FACTOR VEHICLE 1/2/3/4/5 (Facteur Contributif du Véhicule 1/2/3/4/5) : Facteurs contribuant à la collision pour les véhicules désignés.
COLLISION_ID (ID de Collision) : Code unique généré par le système. Clé primaire pour la table des collisions.
VEHICLE TYPE CODE 1/2/3/4/5 (Code de Type de Véhicule 1/2/3/4/5) : Type de véhicule basé sur la catégorie de véhicule sélectionnée (VTT, vélo, voiture/SUV, vélo électrique, trottinette électrique, camion/bus, moto, autre).

IV. Environnement de Développement

Langage: Python

Librairies Utilisées: Selenium, Pandas , BeautifulSoup

Outil: Anaconda jupyter

Navigateur Web: Chrome

V. Détails du Script

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import pandas as pd
from bs4 import BeautifulSoup
import time

url = "https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95/explore/query/SELECT%0A%20%20%60cr

options = Options()
options.add_argument('--headless')

driver = webdriver.Chrome(options=options)

try:
    driver.get(url)
    driver.implicitly_wait(10)

    # Wait for the table with the specified class to be present
    table_locator = (By.TAG_NAME, 'table')
    table = WebDriverWait(driver, 20).until(EC.presence_of_element_located(table_locator))

    if table:
        result_df = pd.DataFrame()
        total_pages = 2000

        for page in range(1, total_pages + 1):
            next_button_locator = (By.CSS_SELECTOR, 'a.next-link')
            next_button = WebDriverWait(driver, 10).until(EC.element_to_be_clickable(next_button_locator))
            driver.execute_script("arguments[0].click();", next_button)
            time.sleep(10)


            table = WebDriverWait(driver, 10).until(EC.presence_of_element_located(table_locator))
            table_html = table.get_attribute('outerHTML')

            # Use BeautifulSoup to extract table data
            soup = BeautifulSoup(table_html, 'html.parser')
            table_data = []
```

```
            table = WebDriverWait(driver, 10).until(EC.presence_of_element_located(table_locator))
            table_html = table.get_attribute('outerHTML')

            # Use BeautifulSoup to extract table data
            soup = BeautifulSoup(table_html, 'html.parser')
            table_data = []

            # Check if table has 'th' (header) and 'td' (data) elements
            if soup.find('th') and soup.find('td'):
                #th = ag-header-cell-comp-wrapper
                for row in soup.find_all('tr'):
                    row_data = [cell.get_text(strip=True) for cell in row.find_all(['th', 'td'])]
                    table_data.append(row_data)

                # Check if table_data has rows
                if table_data:
                    # Convert table data to DataFrame
                    df = pd.DataFrame(table_data[1:], columns=table_data[0])
                    result_df = pd.concat([result_df, df], ignore_index=True)
                    print(result_df.head())
                else:
                    print(f"No data found on page {page}")
            else:
                print(f"No 'th' (header) or 'td' (data) elements found on page {page}")

        result_df.to_excel("nyc_collisions_data_all_pages100K.xlsx", index=False)
        print("Data successfully exported")
    else:
        print("No table")
finally:
    driver.quit()
```

1. Initialisation du Navigateur

– Utilisation de Chrome en mode tête invisible.
– Attente implicite de 10 secondes pour s'assurer que la page est chargée.

2. Navigation sur le Site

– Accès à l'URL du site .
– Utilisation de Selenium pour attendre la présence d'une table spécifique son tag “table”.

3. Scraping des Données

– Itération sur 2000 pages
– Utilisation de Selenium pour cliquer sur le bouton 'Suivant' et changer de page.
– Pause de 10 secondes pour permettre le chargement de la nouvelle page.
– Extraction de la table HTML à l'aide de Selenium et BeautifulSoup.

4. Traitement des Données avec Pandas

– Utilisation de Pandas pour lire la table HTML et la stocker dans un DataFrame.
– Concaténation des DataFrames pour accumuler les données de chaque page.

5. Exportation des Données

– Exportation du DataFrame résultant au format Excel (nyc_collisions_data_all_pages100K.xlsx).

VI. Problèmes Rencontrés et Solutions

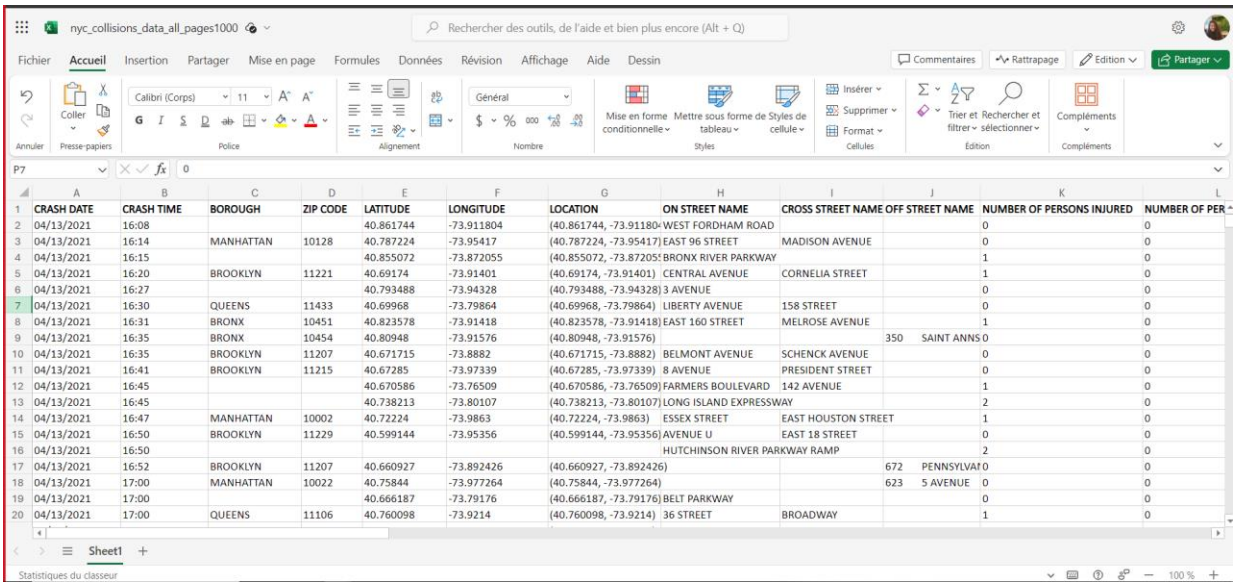
- StaleElementReferenceException
- Utilisation de Selenium pour ré-essayer de localiser les éléments après la détection d'une exception.
- TimeoutException

VII. Améliorations Possibles

- Paramétrage dynamique pour le nombre total de pages.

VIII. Resultat

Fichier avec 100 000 lignes et 29 colonnes :



	A	B	C	D	E	F	G	H	I	J	K	L
1	CRASH DATE	CRASH TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE	LOCATION	ON STREET NAME	CROSS STREET NAME	OFF STREET NAME	NUMBER OF PERSONS INJURED	NUMBER OF PER
2	04/13/2021	16:08			40.861744	-73.911804	(40.861744, -73.91180	WEST FORDHAM ROAD			0	0
3	04/13/2021	16:14	MANHATTAN	10128	40.787224	-73.95417	(40.787224, -73.95417)	EAST 96 STREET	MADISON AVENUE		0	0
4	04/13/2021	16:15			40.855072	-73.872055	(40.855072, -73.87205	BRONX RIVER PARKWAY			1	0
5	04/13/2021	16:20	BROOKLYN	11221	40.69174	-73.91401	(40.69174, -73.91401)	CENTRAL AVENUE	CORNELIA STREET		1	0
6	04/13/2021	16:27			40.793488	-73.94328	(40.793488, -73.94328)	3 AVENUE			0	0
7	04/13/2021	16:30	QUEENS	11433	40.69968	-73.79864	(40.69968, -73.79864)	LIBERTY AVENUE	158 STREET		0	0
8	04/13/2021	16:31	BRONX	10451	40.823578	-73.91418	(40.823578, -73.91418)	EAST 160 STREET	MELROSE AVENUE		1	0
9	04/13/2021	16:35	BRONX	10454	40.80948	-73.91576	(40.80948, -73.91576)			350 SAINT ANNS	0	0
10	04/13/2021	16:35	BROOKLYN	11207	40.671715	-73.8882	(40.671715, -73.8882)	BELMONT AVENUE	SCHENCK AVENUE		0	0
11	04/13/2021	16:41	BROOKLYN	11215	40.67285	-73.97339	(40.67285, -73.97339)	8 AVENUE	PRESIDENT STREET		0	0
12	04/13/2021	16:45			40.670586	-73.76509	(40.670586, -73.76509)	FARMERS BOULEVARD	142 AVENUE		1	0
13	04/13/2021	16:45			40.738213	-73.80107	(40.738213, -73.80107)	LONG ISLAND EXPRESSWAY			2	0
14	04/13/2021	16:47	MANHATTAN	10002	40.72224	-73.9863	(40.72224, -73.9863)	ESSEX STREET	EAST HOUSTON STREET		1	0
15	04/13/2021	16:50	BROOKLYN	11229	40.599144	-73.95356	(40.599144, -73.95356)	AVENUE U	EAST 18 STREET		0	0
16	04/13/2021	16:50						HUTCHINSON RIVER PARKWAY RAMP			2	0
17	04/13/2021	16:52	BROOKLYN	11207	40.660927	-73.892426	(40.660927, -73.892426)			672 PENNSYLVAI	0	0
18	04/13/2021	17:00	MANHATTAN	10022	40.75844	-73.977264	(40.75844, -73.977264)			623 5 AVENUE	0	0
19	04/13/2021	17:00			40.666187	-73.79176	(40.666187, -73.79176)	BELT PARKWAY			0	0
20	04/13/2021	17:00	QUEENS	11106	40.760098	-73.9214	(40.760098, -73.9214)	36 STREET	BROADWAY		1	0

IX. Conclusion

En conclusion, le script de scraping de données mis en place avec Selenium , Pandas et BeautifulSoup a permis de collecter efficacement des informations sur les collisions de véhicules à moteur signalées par la police à New York en 2020 et 2021 . L'utilisation de Selenium a permis d'automatiser la navigation sur le site, tandis que Pandas a facilité le traitement des données tabulaires...

Chapitre 2 : Traitement de données-SSIS

I. Objectif

Le traitement de données avec SQL Server Integration Services (SSIS) a plusieurs objectifs principaux, principalement dans le contexte de l'intégration et de la transformation des données.

II. Outils

SSIS (SQL Server Integration Services) est une plateforme de Microsoft utilisée pour l'intégration des données, le nettoyage des données et le traitement des flux de données dans les solutions de Business Intelligence et de gestion de données. Il permet d'extraire, transformer et charger les données (ETL), d'automatiser les tâches ETL, de maintenir les bases de données et de s'intégrer avec d'autres produits Microsoft pour créer des solutions BI complètes.

III. Réalisation :

1. Composants Utilisés

À l'aide de composants tels que les sources de données, les transformations et les destinations, SSIS facilite l'extraction des données, la transformation selon les besoins, et le chargement dans un emplacement de stockage centralisé :

- Flat File Source : Ce composant est utilisé pour lire des données à partir de fichiers plats, tels que des fichiers texte (CSV, tabulés) ou des fichiers à largeur fixe. Ce composant est couramment utilisé dans le processus ETL (Extract, Transform, Load) pour extraire des données à partir de fichiers plats et les intégrer dans une base de données ou un entrepôt de données.

-Data Conversion (Conversion de données) : Ce composant est utilisé pour convertir les données d'un type de données à un autre. Par exemple, il peut convertir une chaîne de caractères en entier ou une date en format différent. Cela est utile lorsqu'il y a des incompatibilités de types de données entre la source et la destination.

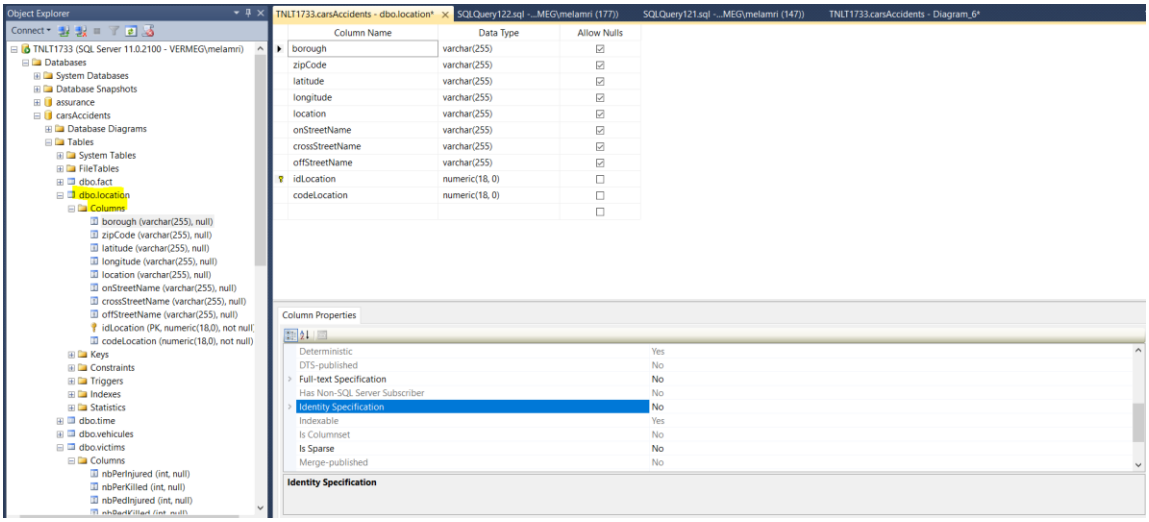
-Lookup : Le composant Lookup est utilisé pour rechercher des valeurs dans une autre table ou source de données. Il est généralement utilisé pour enrichir les données provenant d'une source avec des informations supplémentaires provenant d'une autre source, en utilisant une clé de recherche commune.
-OLE DB Destination : Ce composant est utilisé pour 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 mappings entre les colonnes source et destination.
-OLE DB Command : Le composant OLE DB Command permet d'exécuter des commandes SQL spécifiques pour chaque ligne de données. Il est utile lorsque vous avez besoin de manipuler les données ou d'effectuer des opérations plus complexes sur une base de ligne par ligne, par exemple, mettre à jour des valeurs, exécuter des procédures stockées, etc.

2. Démarche :

Étape 1 :

Dans cette étape, créez des tables de destination de données pour les dimensions et les faits. Nous allons créer 4 tables de dimension et 1 table de fait pour charger les données dans le datawarehouse provenant des fichiers flat file sources.

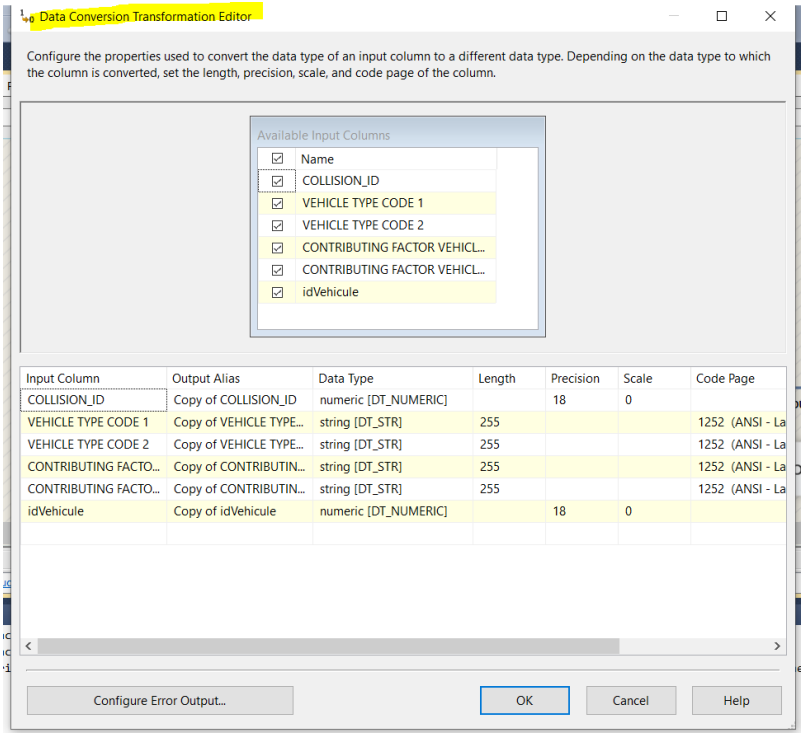
Remarque : Le datawarehouse est sur SQL SERVER.



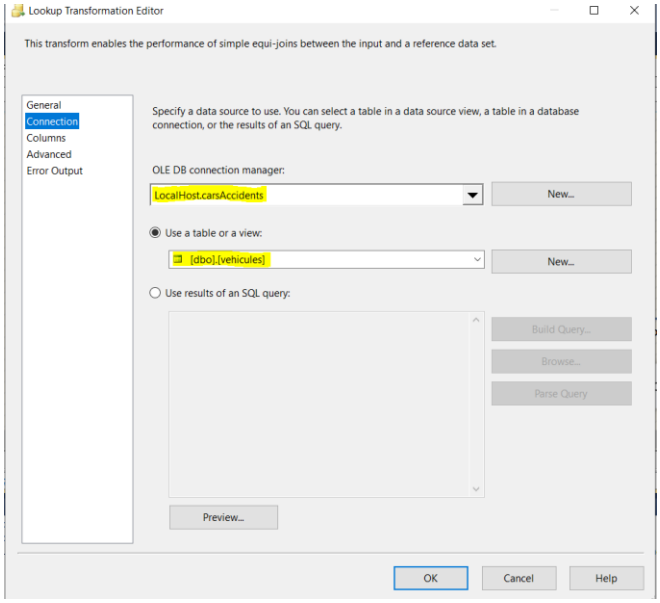
Étape 2 :

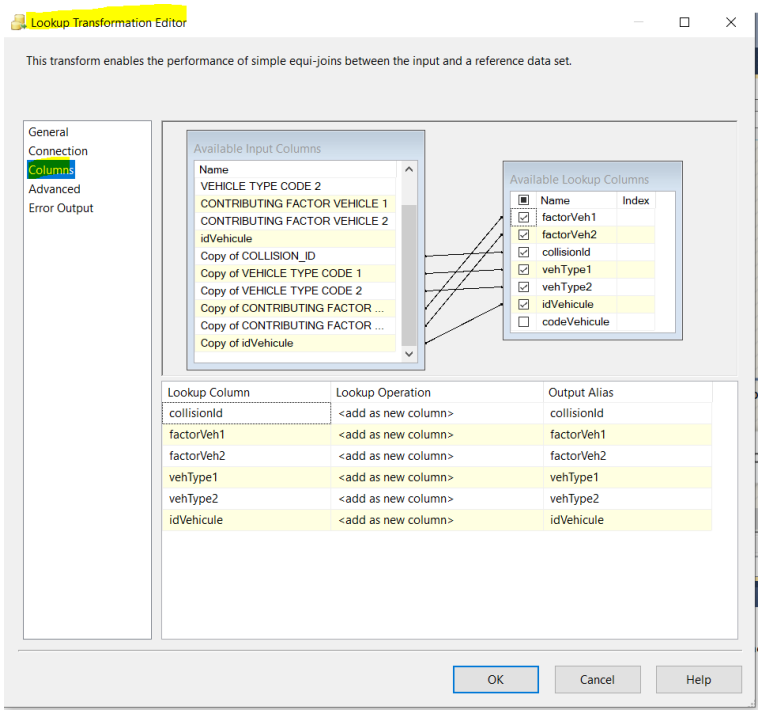
Faites glisser et déposez data flow dans control flow task -> accédez à data flow et faites glisser et déposez le flat file source dont notre données extraites et enregistrées dans un fichier text

Utiliser le composant **“Data Conversion”** pour avoir les types convenables avec notre data et les colonnes crées dan sql server

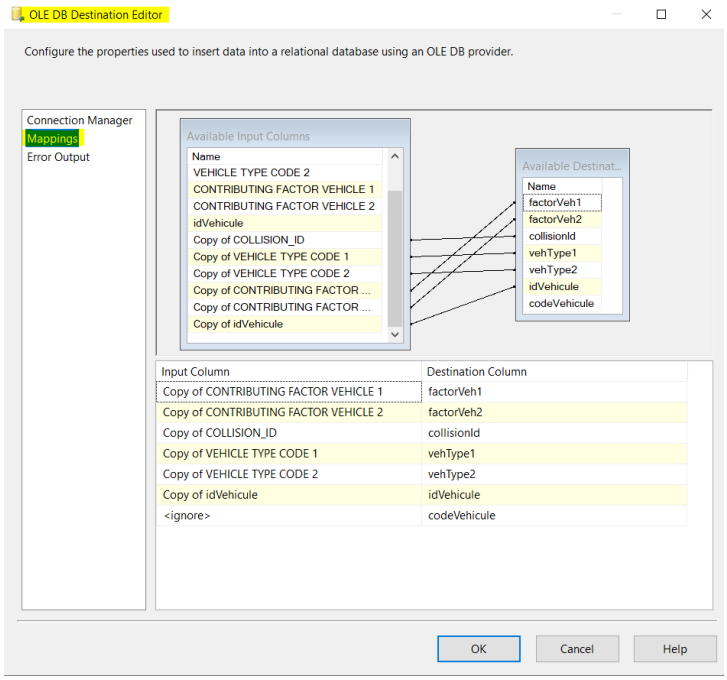


Utiliser le composant **“Lookup”** pour correspondre les colonnes existantes dans le flat file et les colonnes de la table crées dans sql server



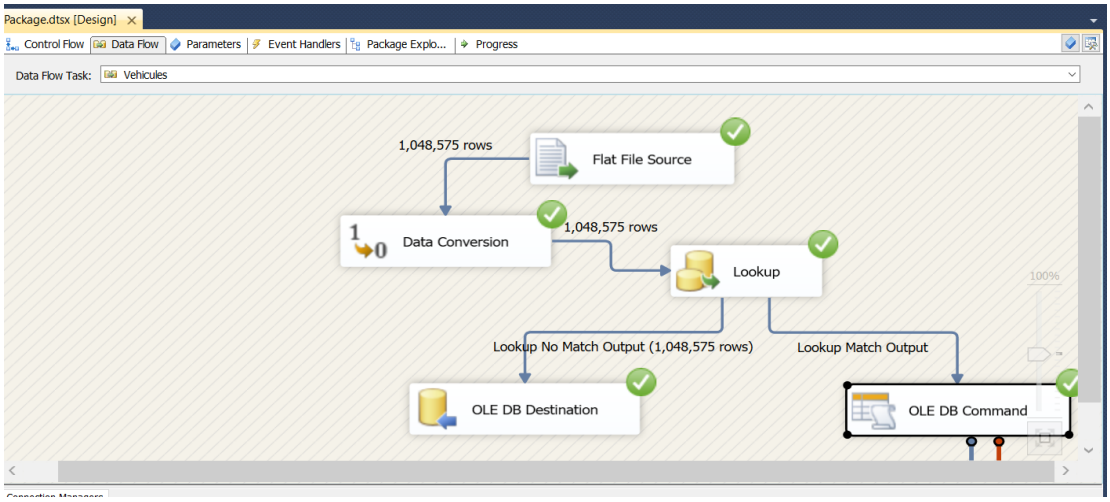


Ajouter le composant **“OLE DB Destination”** pour charger nos données dans notre base de données “carAccidents” en SQL server pour bien spécifier la table de destination et les mappings entre les colonnes source et destination.



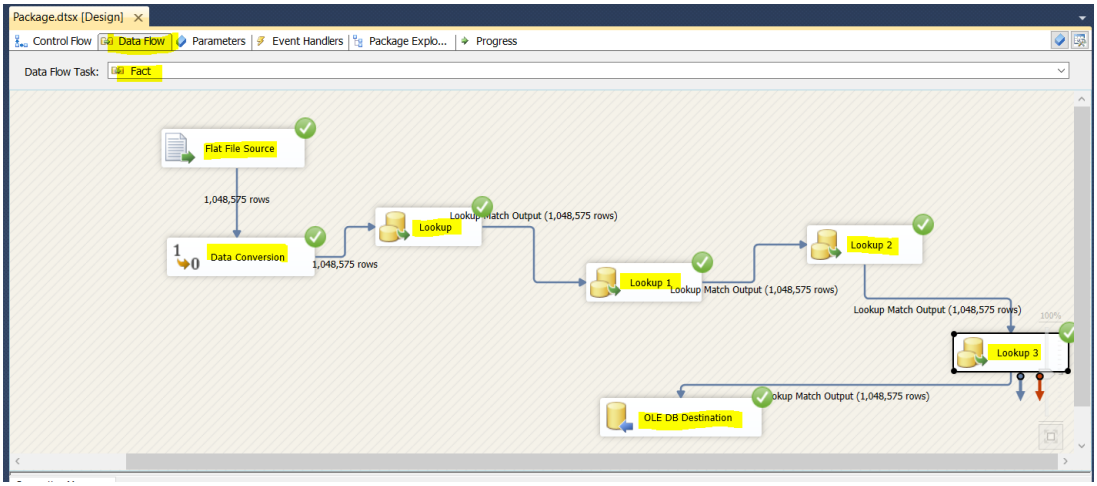
Ajouter le composant **“OLE DB Command”** pour executer la comande sql de mise à jour des valeurs si existantes

Refaire cette étape pour chacune des tables crées vehicules , location , time et victims



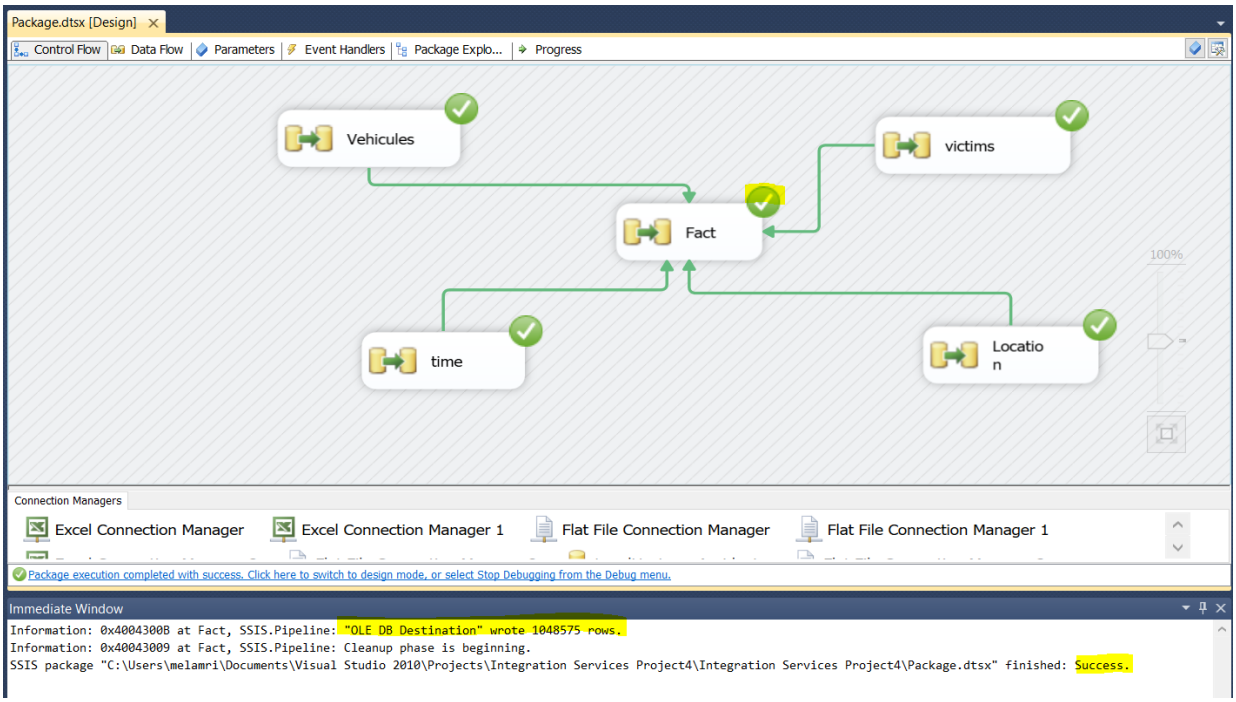
Étape 3 :

Refaire l'étape 2 pour créer la table de fait qui contient les clés primaires de nos tables vehicules , location , time et victims en utilisant des look up pour chacune

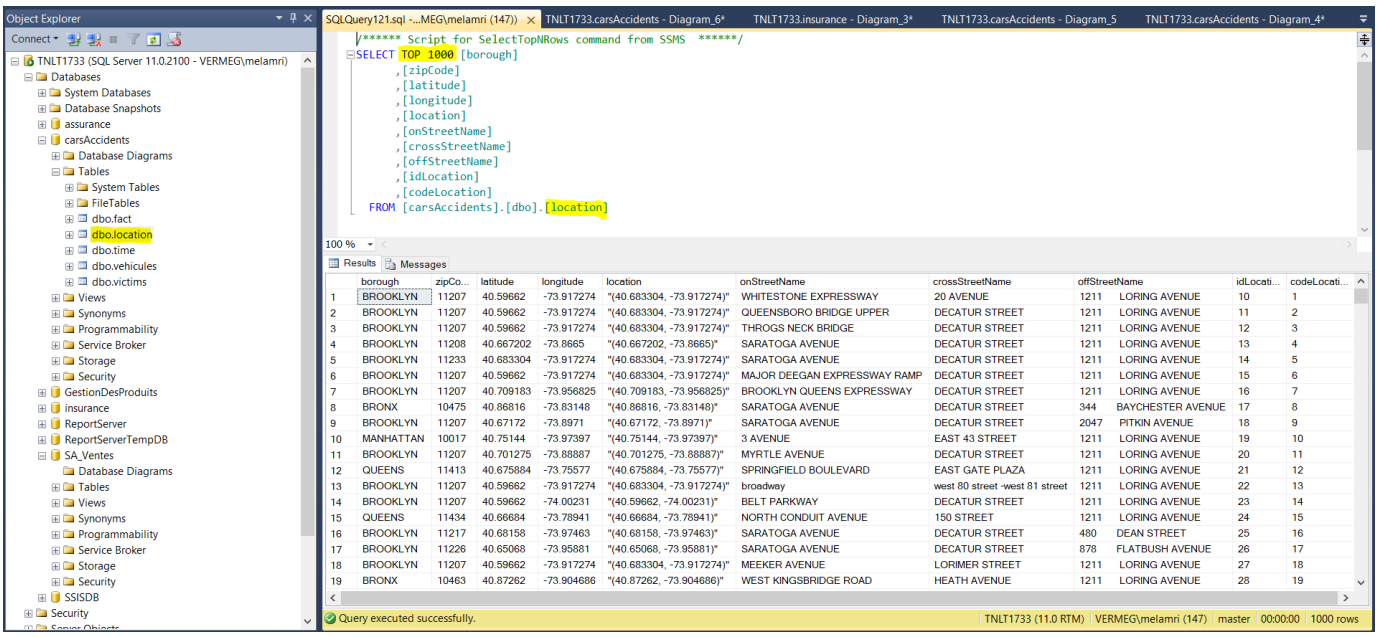


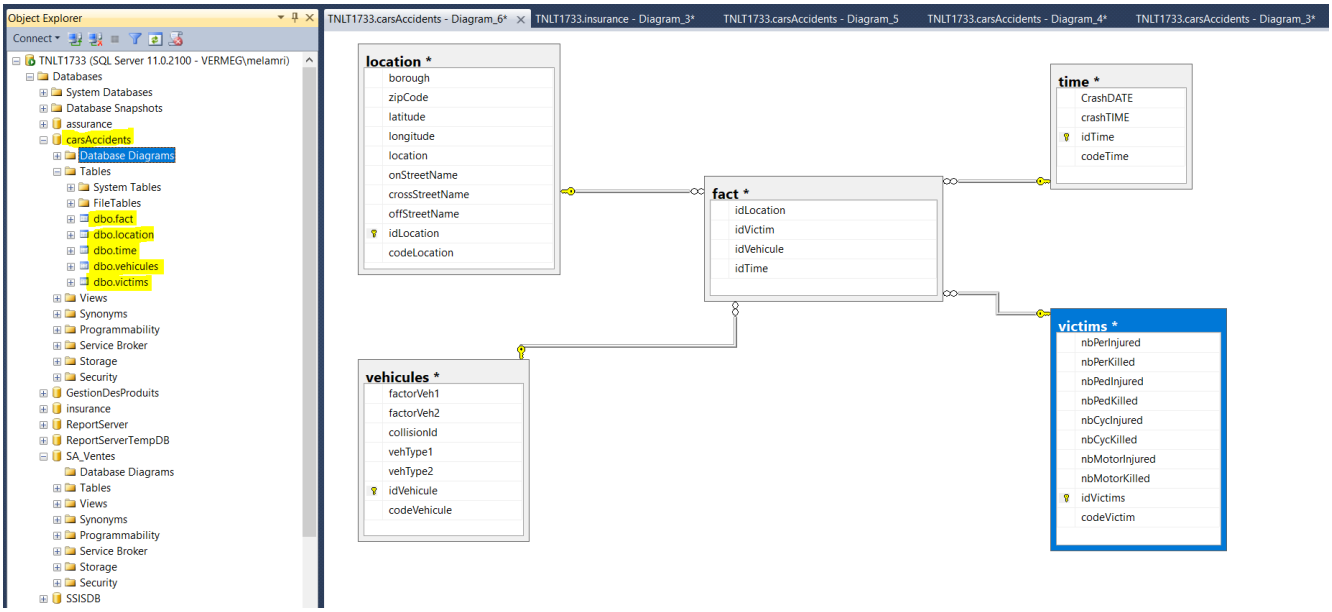
Étape 4 :

Lancer l'exécution de nos data flow et charger les données dans la base de données “carAccidents” dans sql server



** Exemple de Chargement des données dans la table location dans la BD "carAccidents"*





3. Conclusion :

En conclusion , Avec SSIS on a utilisé Data Conversion pour convertir les types de données, Lookup pour la recherche des valeurs dans d'autres sources, OLE DB Destination pour charger des données dans une destination OLE DB, et OLE DB Command pour exécuter des commandes SQL sur les données. En faite , ces composants ont facilité l'extraction, la transformation et le chargement (ETL) des données dans la solutions SSIS afin de l'utiliser dans l'etape suivante pour préparer les dashboard avec PowerBI