2024

Analysis and Visualization of Road Traffic Accidents

Data science Project – Projet 2

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

```python
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 = []
```

```python
            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

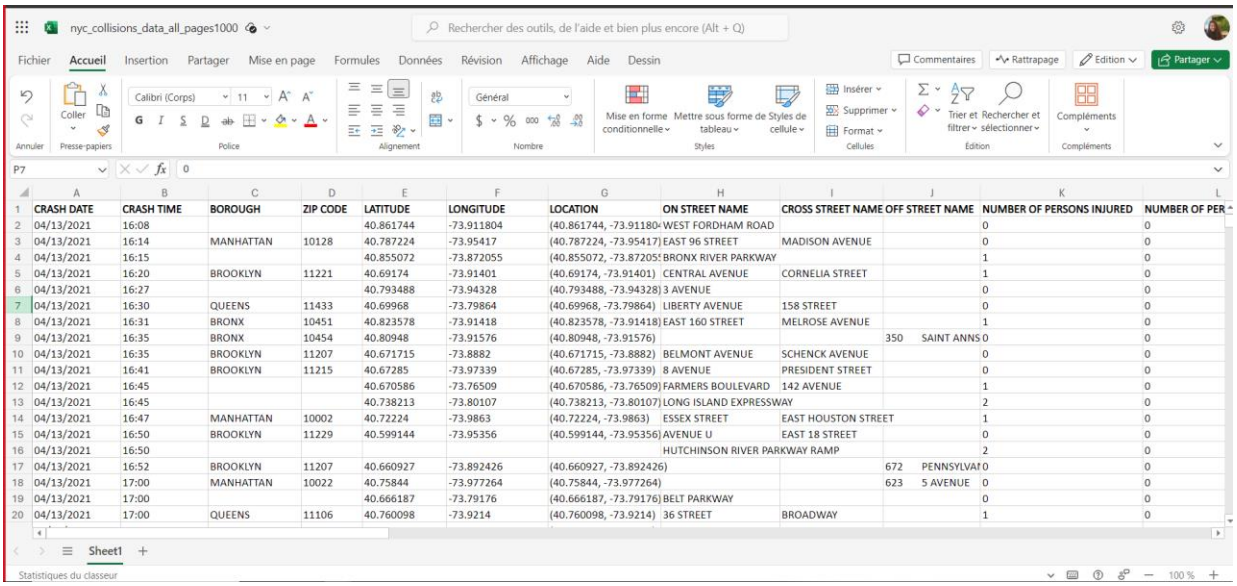
- StaleElementReferenceException
- Utilisation de Selenium pour ré-essayer de localiser les éléments après la détection d'une exception.
- TimeoutException

VII. Améliorations Possibles

- Paramétrage dynamique pour le nombre total de pages.

VIII. Resultat

Fichier avec 100 000 lignes et 29 colonnes :



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
04/13/2021	16:08			40.861744	-73.911804	(40.861744, -73.91180	WEST FORDHAM ROAD			0	0
04/13/2021	16:14	MANHATTAN	10128	40.787224	-73.95417	(40.787224, -73.95417)	EAST 96 STREET	MADISON AVENUE		0	0
04/13/2021	16:15			40.855072	-73.872055	(40.855072, -73.87205	BRONX RIVER PARKWAY			1	0
04/13/2021	16:20	BROOKLYN	11221	40.69174	-73.91401	(40.69174, -73.91401)	CENTRAL AVENUE	CORNELIA STREET		1	0
04/13/2021	16:27			40.793488	-73.94328	(40.793488, -73.94328)	3 AVENUE			0	0
04/13/2021	16:30	QUEENS	11433	40.69968	-73.79864	(40.69968, -73.79864)	LIBERTY AVENUE	158 STREET		0	0
04/13/2021	16:31	BRONX	10451	40.823578	-73.91418	(40.823578, -73.91418)	EAST 160 STREET	MELROSE AVENUE		1	0
04/13/2021	16:35	BRONX	10454	40.80948	-73.91576	(40.80948, -73.91576)			350 SAINT ANNS	0	0
04/13/2021	16:35	BROOKLYN	11207	40.671715	-73.8882	(40.671715, -73.8882)	BELMONT AVENUE	SCHENCK AVENUE		0	0
04/13/2021	16:41	BROOKLYN	11215	40.67285	-73.97339	(40.67285, -73.97339)	8 AVENUE	PRESIDENT STREET		0	0
04/13/2021	16:45			40.670586	-73.76509	(40.670586, -73.76509)	FARMERS BOULEVARD	142 AVENUE		1	0
04/13/2021	16:45			40.738213	-73.80107	(40.738213, -73.80107)	LONG ISLAND EXPRESSWAY			2	0
04/13/2021	16:47	MANHATTAN	10002	40.72224	-73.9863	(40.72224, -73.9863)	ESSEX STREET	EAST HOUSTON STREET		1	0
04/13/2021	16:50	BROOKLYN	11229	40.599144	-73.95356	(40.599144, -73.95356)	AVENUE U	EAST 18 STREET		0	0
04/13/2021	16:50						HUTCHINSON RIVER PARKWAY RAMP			2	0
04/13/2021	16:52	BROOKLYN	11207	40.660927	-73.892426	(40.660927, -73.892426)			672 PENNSYLVAI	0	0
04/13/2021	17:00	MANHATTAN	10022	40.75844	-73.977264	(40.75844, -73.977264)			623 5 AVENUE	0	0
04/13/2021	17:00			40.666187	-73.79176	(40.666187, -73.79176)	BELT PARKWAY			0	0
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.

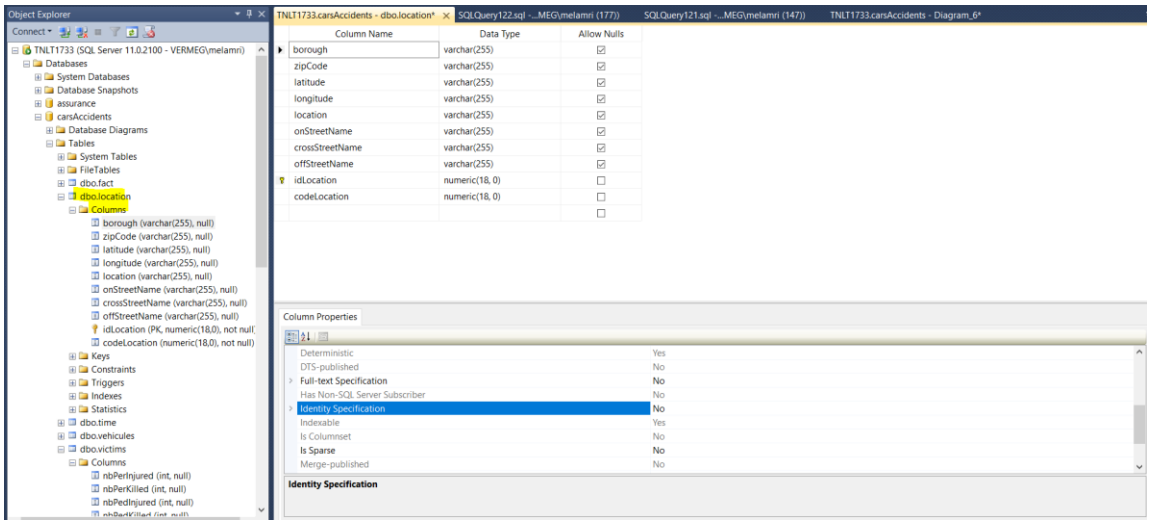
-Data Conversion (Conversion de données) : Ce composant est utilisé pour convertir les données d'un type de données à un autre. Par exemple, il peut convertir une chaîne de caractères en entier ou une date en format différent. Cela est utile lorsqu'il y a des incompatibilités de types de données entre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

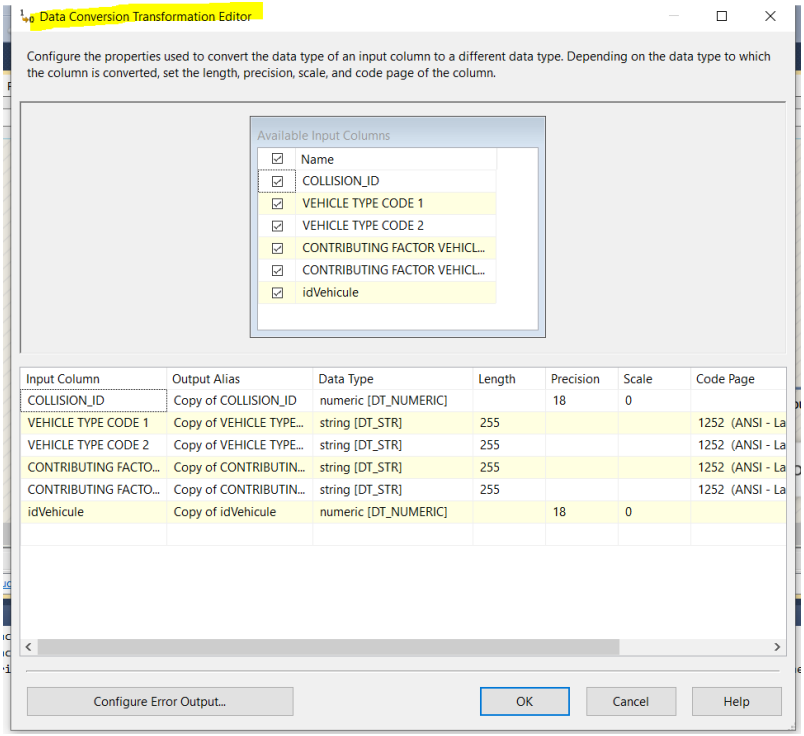
Remarque : Le datawarehouse est sur SQL SERVER.



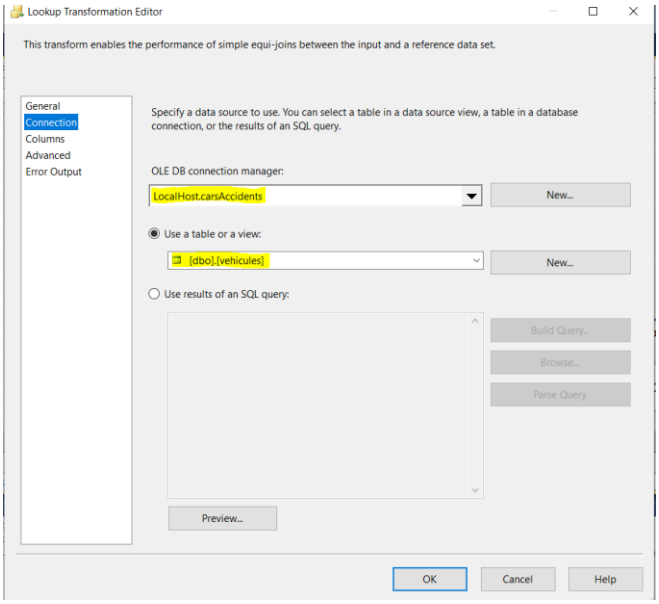
Étape 2 :

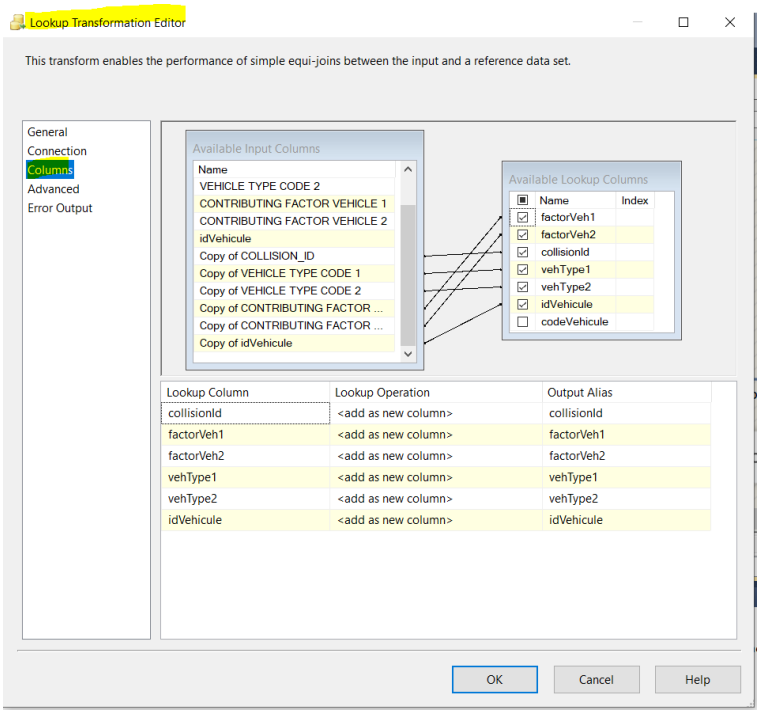
Faites glisser et déposez data flow dans control flow task -> accédez à data flow et faites glisser et déposez le flat file source dont notre données extraites et enregistrées dans un fichier text

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

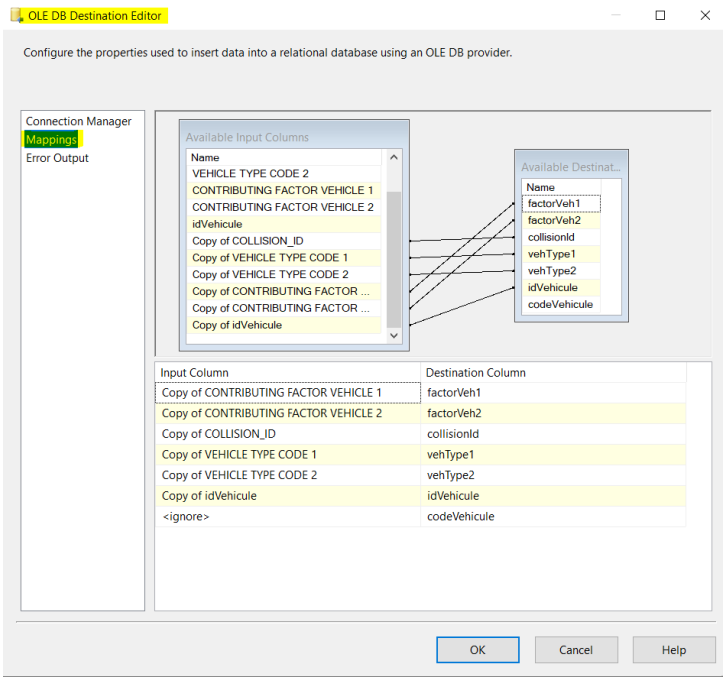


Utiliser le composant **“Lookup”** pour correspondre les colonnes existantes dans le flat file et les colonnes de la table crées dans sql server



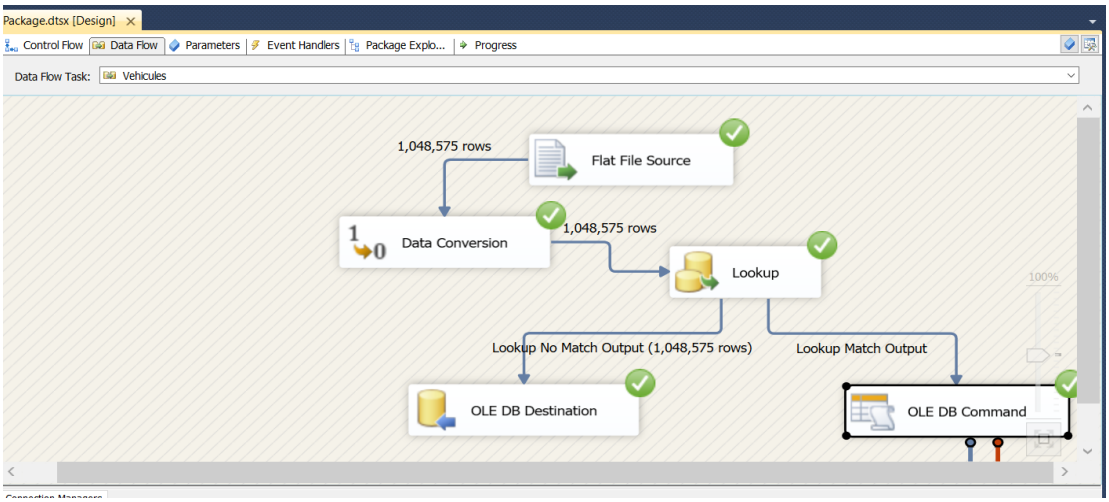


Ajouter le composant **“OLE DB Destination”** pour charger nos données dans notre base de données “carAccidents” en SQL server pour bien spécifier la table de destination et les mappings entre les colonnes source et destination.



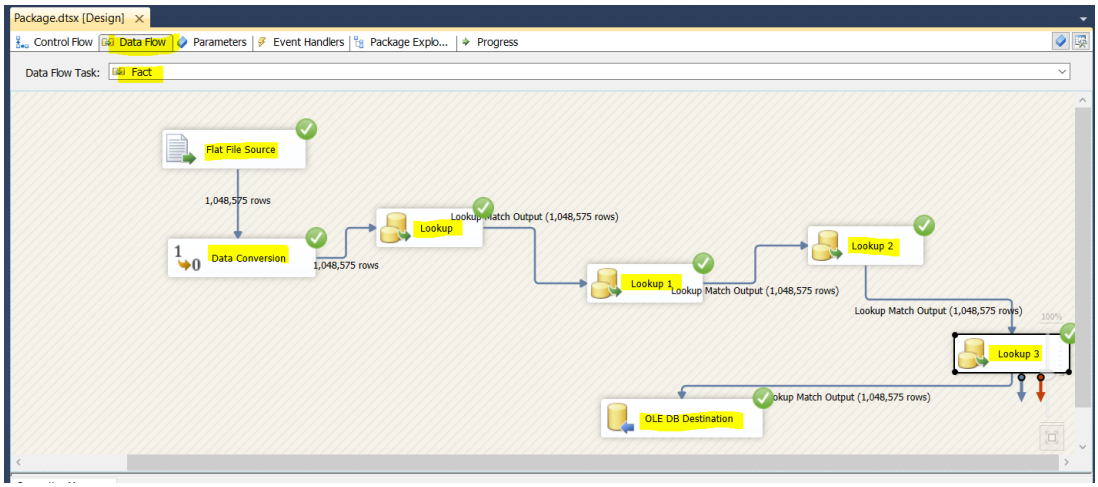
Ajouter le composant **“OLE DB Command**” pour executer la comande sql de mise à jour des valeurs si existantes

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



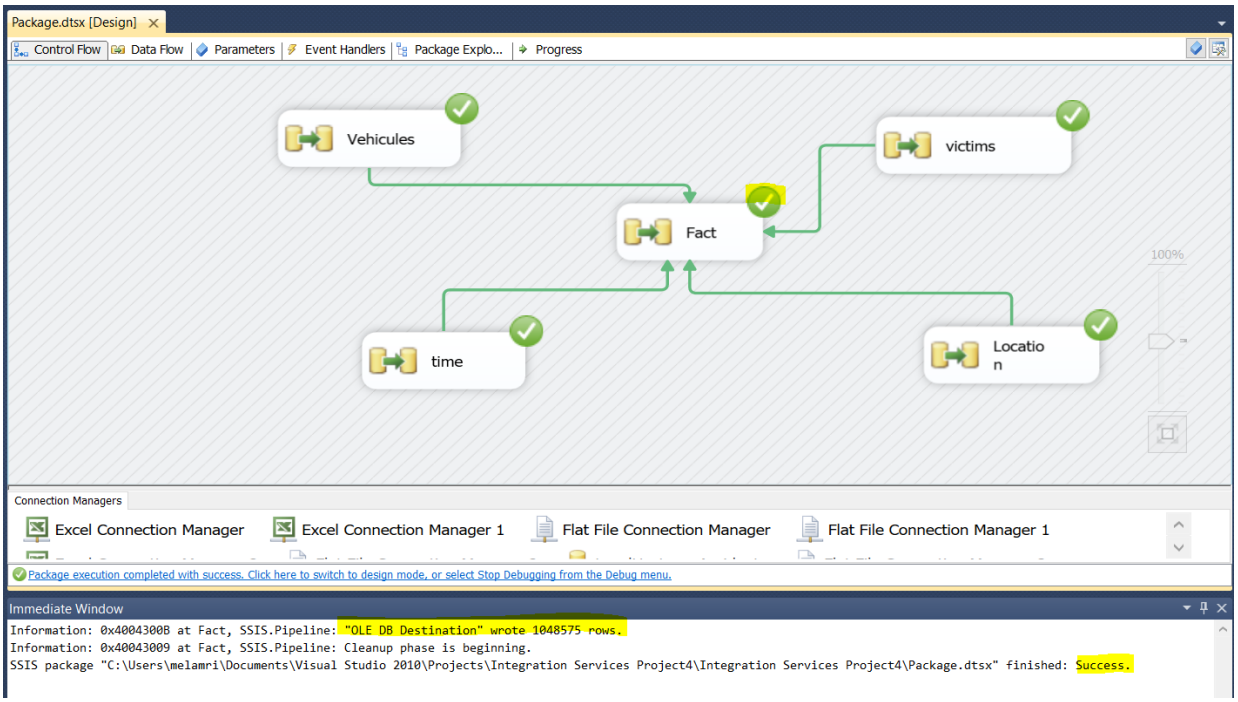
Étape 3 :

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



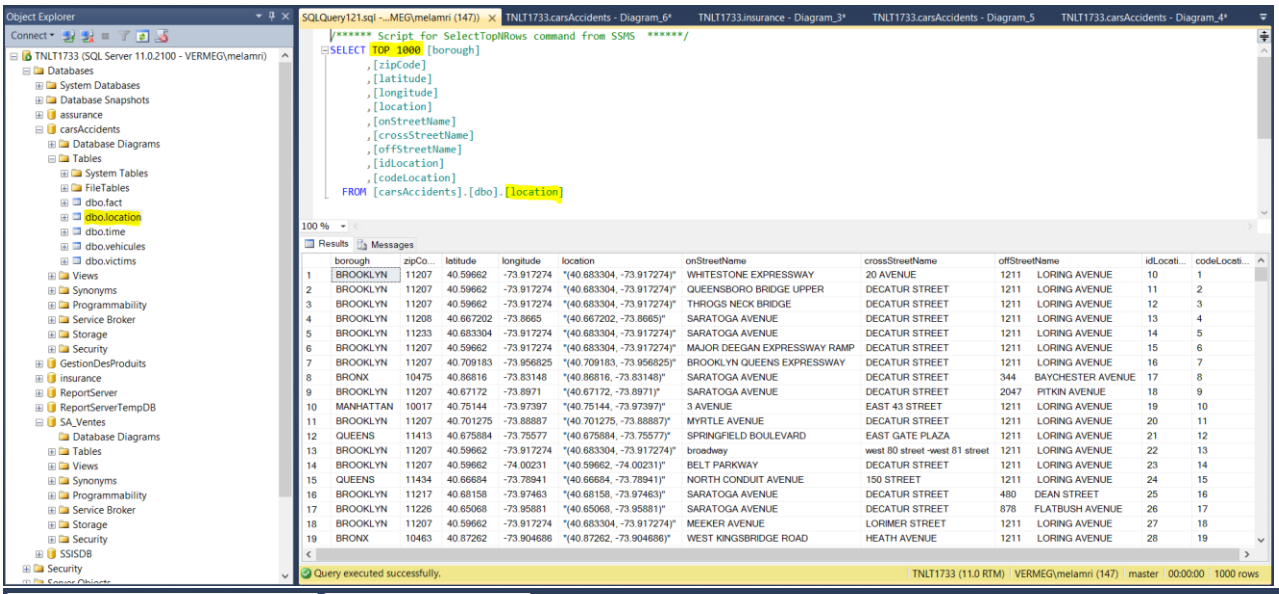
Étape 4 :

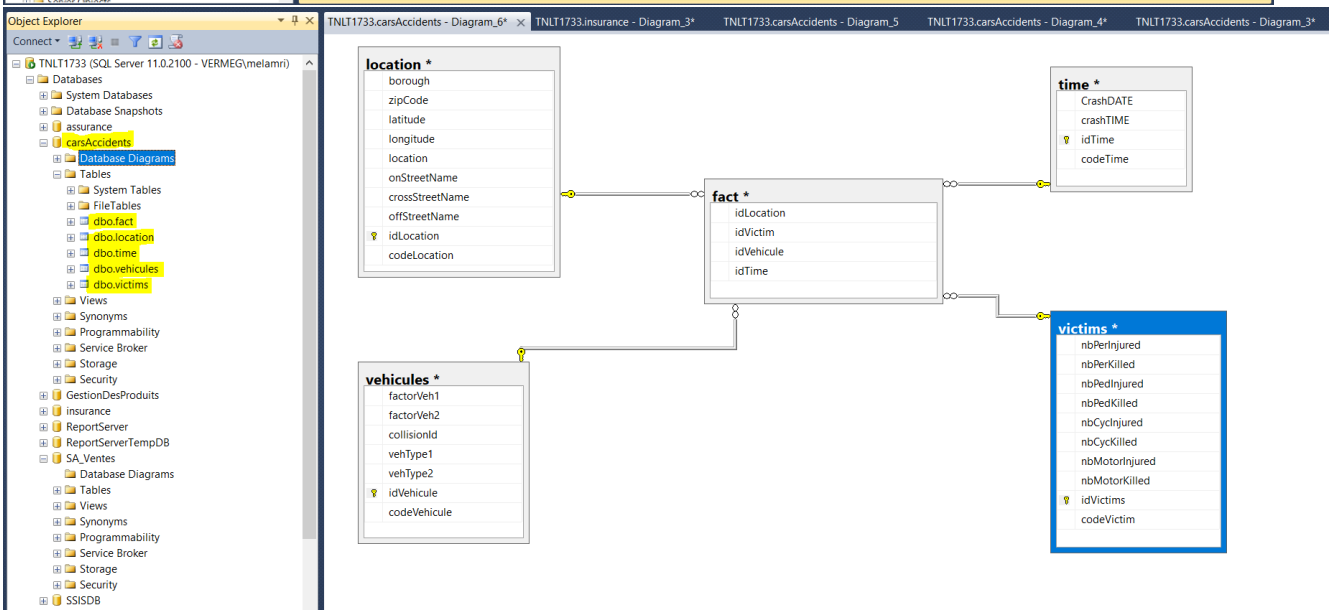
Lancer l'exécution de nos data flow et charger les données dans la base de données “carAccidents” dans sql server



*

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

Chapitre 3 : Analyse de données-PowerBI

I. Objectif

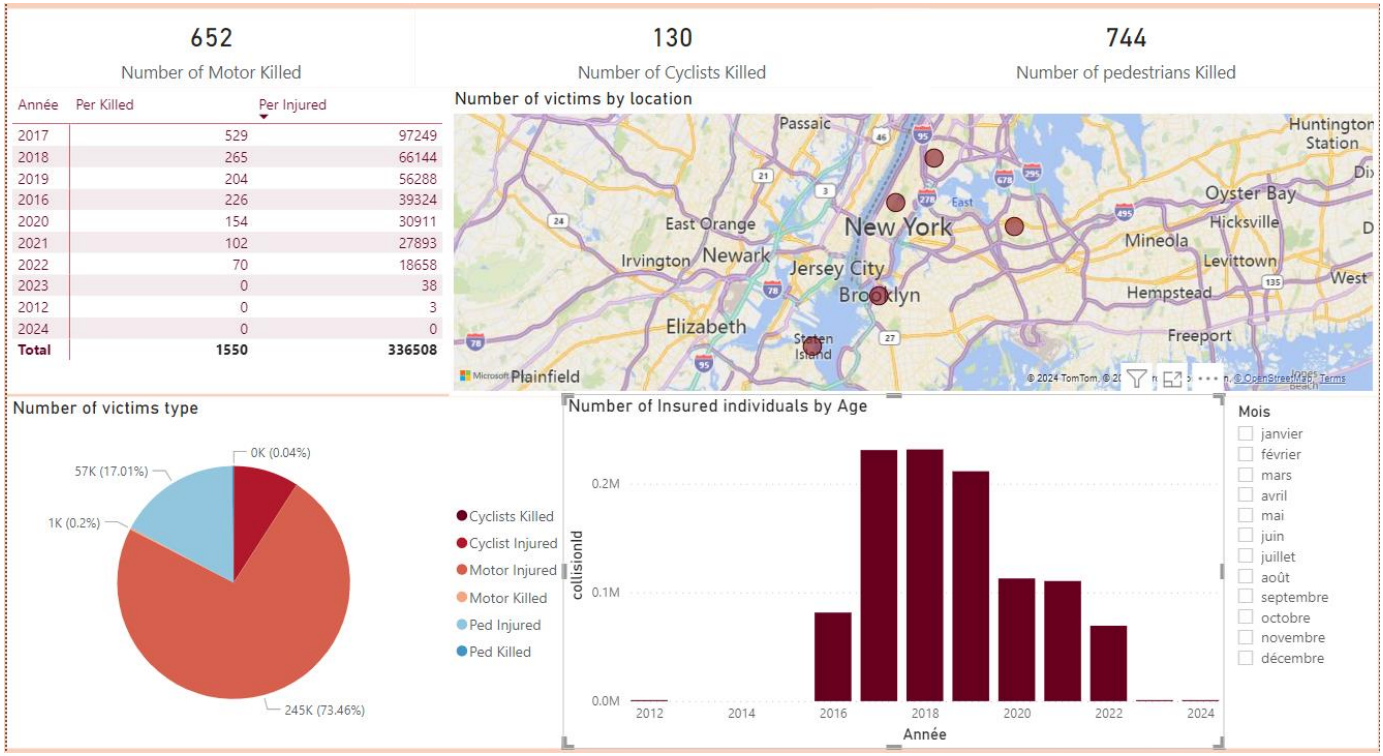
L'utilisation de Power BI pour analyser les données d'accidents à New York vise à fournir une plateforme puissante pour la visualisation, l'analyse et la compréhension des données liées aux accidents.

II. Outils

Power BI est un outil de Business Intelligence (BI) développé par Microsoft. Il offre une suite de services, d'applications et de connecteurs qui permettent aux utilisateurs de visualiser et d'analyser leurs données de manière interactive.
Il offre une approche intégrée, de l'extraction des données à la création de rapports interactifs. Les fonctionnalités de visualisation avancées, de modélisation de données et de partage facilitent la compréhension des données et la prise de décision basée sur des informations pertinentes.

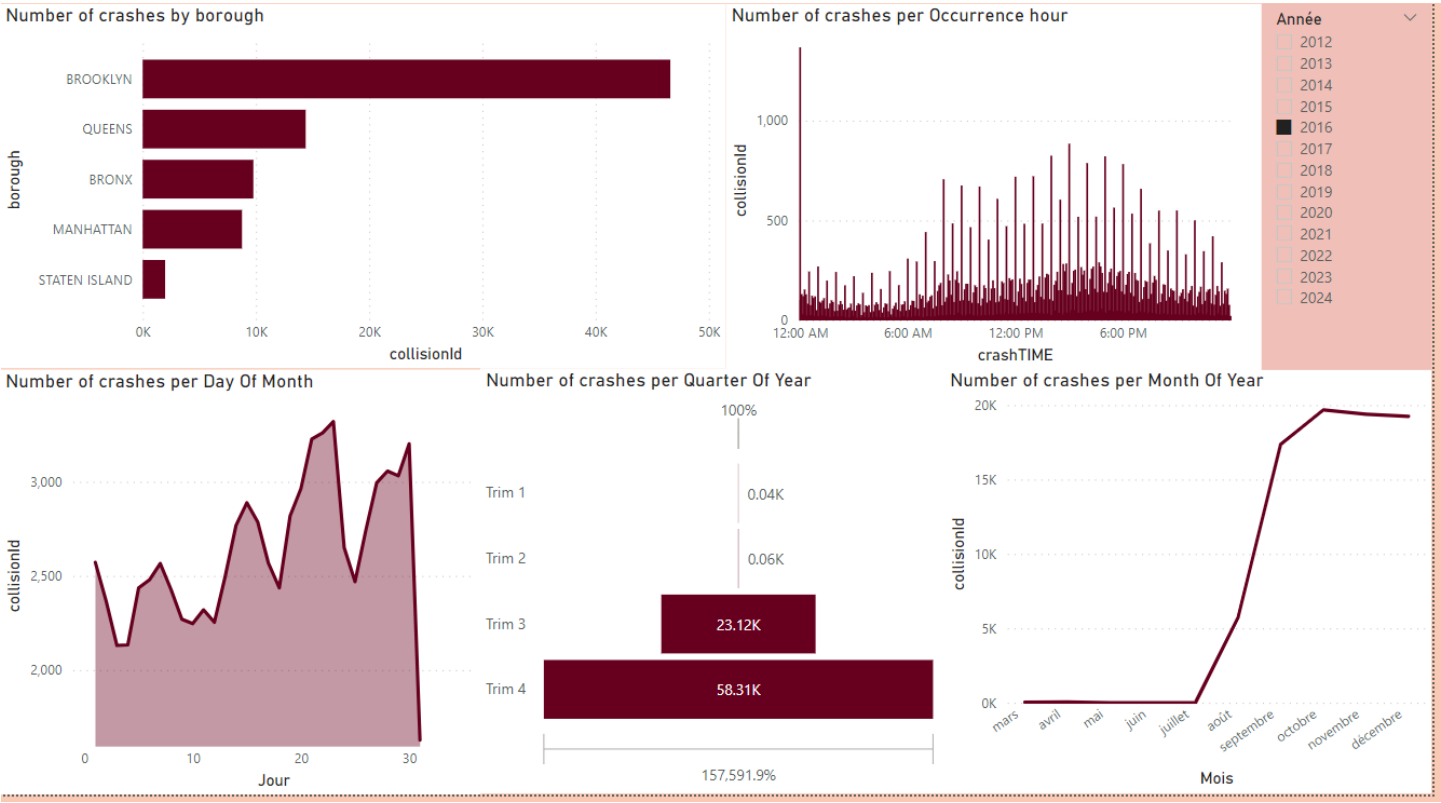
III. Réalisation :

1. Page 1 - Tableau de bord principal



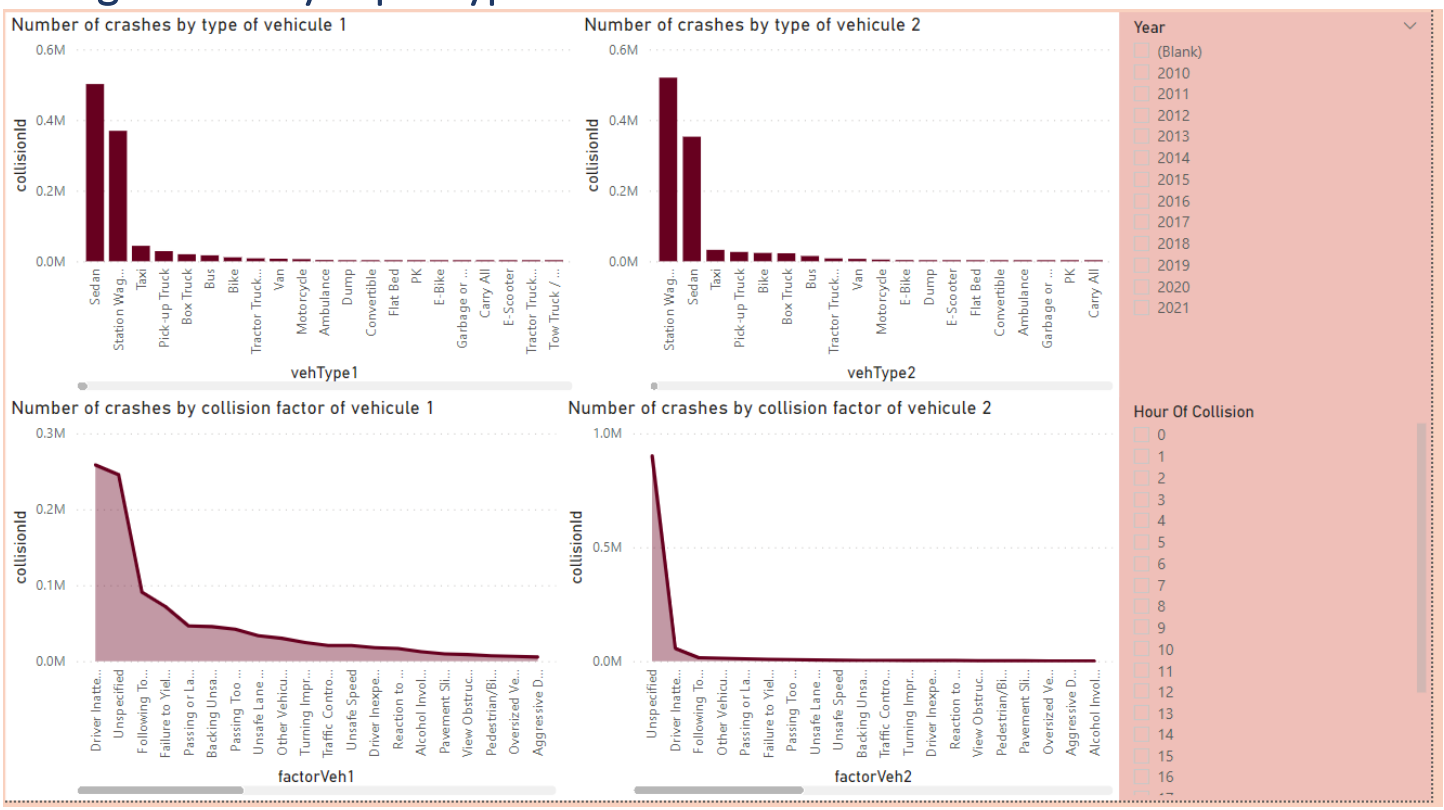
1. **<u>Chiffres des accidents :</u>**
   - On a inclus des indicateurs pour les accidents impliquant des cyclistes tués, des piétons tués et des personnes tuées dans des véhicules motorisés. Ces chiffres donnent une vue d'ensemble de l'ampleur des accidents dans la ville.
2. **<u>Carte interactive :</u>**
   - On a intégré une carte interactive montrant les zones de New York City où les accidents se sont produits, divisées par les cinq arrondissements (Bronx, Queens, Manhattan, Brooklyn et Staten Island).
3. **<u>Pie Chart (Diagramme circulaire) pour le nombre de victimes par type :</u>**
   - Le Pie Chart présente une répartition visuelle des victimes d'accidents par type, y compris les personnes tuées et blessées dans des véhicules motorisés, les cyclistes tués et blessés, ainsi que les piétons tués et blessés. Cette représentation permet une visualisation rapide des proportions relatives de chaque catégorie de victimes.
4. **<u>Stacked Column Chart (Graphique à colonnes empilées) pour le nombre d'accidents par année :</u>**
   - Le Stacked Column Chart illustre la tendance du nombre total d'accidents par année, couvrant la période de 2012 à 2024. Cette représentation graphique permet de visualiser les variations annuelles et de comparer facilement les volumes d'accidents sur une période prolongée.
5. **<u>Tableau Matrix (Tableau matriciel) pour les nombres de personnes tuées par année avec filtre interactif :</u>**
   - Le Tableau Matrix illustre les nombres de personnes tuées au cours des années 2012 à 2024. Ce tableau agit également comme un filtre interactif pour contrôler l'ensemble du dashboard en fonction des années sélectionnées.
6. **<u>Filtres interactifs :</u>**
   - On a inclus un filtre interactif pour permettre aux utilisateurs de sélectionner les mois de l'année, offrant ainsi la possibilité d'explorer les données de manière plus granulaire.

2. Page 2 - analyse spatio temporel



1. **<u>Clustered Bar Chart (Diagramme à barres groupées) :</u>**
   - Ce graphique présente le nombre d'accidents par arrondissement, offrant une comparaison visuelle de la fréquence des accidents dans chacun des cinq arrondissements de New York City.
2. **<u>Stacked Column Chart (Diagramme à colonnes empilées) :</u>**
   - Ce graphique illustre le nombre d'accidents par heure d'occurrence, montrant la répartition des accidents tout au long de la journée.
3. **<u>Area Chart (Graphique en aire) :</u>**
   - Cet graphique affiche le nombre d'accidents par jour du mois, ce qui permet de visualiser les tendances mensuelles dans les accidents de véhicules.
4. **<u>Funnel Chart (Diagramme en entonnoir) :</u>**
   - Ce graphique montre le nombre d'accidents par trimestre de l'année, offrant une perspective sur la répartition des accidents sur l'ensemble de l'année.
5. **<u>Line Chart (Graphique en ligne) :</u>**
   - Ce graphique présente le nombre d'accidents par mois de l'année, permettant de mettre en évidence les variations saisonnières et les tendances annuelles.
6. **<u>Filtre interactif :</u>**
   - On a inclus un filtre interactif permettant aux utilisateurs de sélectionner les années entre 2012 et 2024, offrant ainsi la possibilité d'explorer les données sur une plage temporelle spécifique.

3. Page 3 - Analyse par type et facteur de vehicules



1. **<u>Line and Clustered Column Chart (Graphique linéaire et en colonnes groupées) - Nombre d'accidents par type de véhicule 1 :</u>**
   - Ce graphique présente le nombre d'accidents impliquant différents types de véhicules de type 1 (par exemple, taxi, berline, VUS, vélo, autobus, camion, etc.) au fil du temps, permettant de visualiser les tendances et les variations dans les accidents par type de véhicule 1.
2. **<u>Line and Clustered Column Chart (Graphique linéaire et en colonnes groupées) - Nombre d'accidents par type de véhicule 2 :</u>**
   - Ce graphique illustre le nombre d'accidents impliquant différents types de véhicules de type 2 (comme décrit ci-dessus) au fil du temps, offrant une comparaison visuelle des accidents par type de véhicule 2.
3. **<u>Stacked Area Chart (Graphique en aire empilée) - Nombre d'accidents par facteur de collision pour le véhicule 1 :</u>**
   - Ce graphique présente le nombre d'accidents où différents facteurs de collision ont été impliqués pour le véhicule 1 (par exemple, inattention, distraction, suivi trop serré, changement de voie dangereux, non-respect du droit de passage, etc.), permettant de visualiser les causes principales des accidents impliquant ce type de véhicule.
4. **<u>Filtre interactif - Années de 2012 à 2024 :</u>**
   - On a inclus un filtre interactif permettant aux utilisateurs de sélectionner une plage d'années spécifique, ce qui permet d'explorer les données sur une période donnée.
5. **<u>Filtre interactif - 24 heures :</u>**
   - On a également inclus un filtre interactif pour les 24 heures de la journée, ce qui permet de filtrer les données en fonction de l'heure de la journée où les accidents se sont produits.

IV. Conclusion :

En utilisant Power BI, On a pu créer un tableau de bord interactif qui permet aux utilisateurs d'explorer et d'analyser les données sur les accidents de véhicules à New York City de manière efficace et visuelle. Cette approche facilite une compréhension approfondie des données et favorise la prise de décisions éclairées.

Chapitre 4 –Machine Learning

I. Analyse des Accidents de Voiture et Modélisation Prédictive

1. Introduction

Les accidents de voiture sont malheureusement fréquents et peuvent avoir des conséquences dévastatrices. Dans ce rapport, nous allons examiner un ensemble de données sur les accidents de voiture et utiliser des techniques d'analyse de données et de modélisation prédictive pour comprendre les tendances et développer un modèle capable de prédire le nombre d'accidents.

2. Méthodologie

1. Chargement et Prétraitement des Données :

Nous avons commencé par charger les données à partir d'un fichier Excel contenant des informations sur les accidents de voiture. Ensuite, nous avons converti la colonne 'CRASH DATE' en format de date et extrait des caractéristiques supplémentaires telles que le jour de la semaine, le mois et l'année.

2. Analyse Exploratoire des Données :

Nous avons analysé les données en regroupant les accidents par date et en comptant le nombre d'accidents sur chaque date. Ensuite, nous avons visualisé la distribution du nombre d'accidents à l'aide d'un histogramme et observé comment le nombre d'accidents varie au fil du temps à l'aide d'un graphique linéaire.

Nous avons également examiné la variation du nombre d'accidents par mois et par jour de la semaine en utilisant des graphiques à barres. De plus, nous avons exploré la relation entre le jour de la semaine et le nombre d'accidents à l'aide d'un nuage de points.

3. Modélisation Prédictive

Nous avons utilisé un modèle de régression de forêt aléatoire pour prédire le nombre d'accidents. Après avoir entraîné le modèle sur les données historiques, nous l'avons utilisé pour prédire le nombre d'accidents pour une date spécifique.

3. Outils / Librairies :

Algorithme :

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor

# Load the dataset
data = pd.read_excel('C:/Users/ssghaier/Documents/.machine learning/Motor_Vehicle_Collisions_-_Crashes_20240131_cleaned.xlsx')

# Convert 'CRASH DATE' column to datetime format
data['CRASH DATE'] = pd.to_datetime(data['CRASH DATE'], format='%m/%d/%Y')

# Extract additional features from the date
data['DAY_OF_WEEK'] = data['CRASH DATE'].dt.dayofweek
data['MONTH'] = data['CRASH DATE'].dt.month
data['YEAR'] = data['CRASH DATE'].dt.year

# Group data by date and count the number of accidents on each date
accidents_per_date = data.groupby('CRASH DATE').size().reset_index(name='ACCIDENTS_COUNT')

# Define features (X) and target variable (y)
X = accidents_per_date[['CRASH DATE']]  # Only the date itself
X['DAY_OF_WEEK'] = X['CRASH DATE'].dt.dayofweek
X['MONTH'] = X['CRASH DATE'].dt.month
X['YEAR'] = X['CRASH DATE'].dt.year
X = X.drop(columns=['CRASH DATE'])  # Drop the original date column
y = accidents_per_date['ACCIDENTS_COUNT']

# Initialize Random Forest Regressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
rf_regressor.fit(X, y)

# Prepare input data for prediction (March 16, 2016)
date_to_predict = pd.to_datetime('2026-03-16')  # March 16, 2016
input_data = pd.DataFrame({'DAY_OF_WEEK': [date_to_predict.dayofweek],
                           'MONTH': [date_to_predict.month],
                           'YEAR': [date_to_predict.year]})

# Make prediction
predicted_accidents = rf_regressor.predict(input_data)

# Round the predicted value to the nearest integer
predicted_accidents_integer = int(round(predicted_accidents[0]))

print("Predicted number of accidents on March 16, 2026:", predicted_accidents_integer)
```

```
# Plot histogram
plt.hist(accidents_per_date['ACCIDENTS_COUNT'], bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Number of Accidents')
plt.ylabel('Frequency')
plt.title('Distribution of Number of Accidents')
plt.show()

# Plot line graph
plt.plot(accidents_per_date['CRASH DATE'], accidents_per_date['ACCIDENTS_COUNT'], marker='o', linestyle='-')
plt.xlabel('Date')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents Over Time')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plot bar graph by month
accidents_per_date['MONTH'] = accidents_per_date['CRASH DATE'].dt.month

sns.barplot(x='MONTH', y='ACCIDENTS_COUNT', data=accidents_per_date, palette='viridis')
plt.xlabel('Month')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents by Month')
plt.show()

# Plot bar graph by day of the week
accidents_per_date['DAY_OF_WEEK'] = accidents_per_date['CRASH DATE'].dt.dayofweek
sns.barplot(x='DAY_OF_WEEK', y='ACCIDENTS_COUNT', data=accidents_per_date, palette='magma')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
plt.title('Number of Accidents by Day of the Week')
plt.show()

# Plot scatter plot
plt.scatter(x='DAY_OF_WEEK', y='ACCIDENTS_COUNT', data=accidents_per_date, color='green')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Accidents')
plt.title('Scatter Plot of Number of Accidents by Day of the Week')
plt.show()
```

Ce code effectue plusieurs analyses des données sur les accidents de voiture en utilisant Python et plusieurs bibliothèques, notamment Pandas, Matplotlib, Seaborn ...

Voici une explication détaillée de ce que fait chaque partie du code :

1. Importation des bibliothèques nécessaires : pandas pour la manipulation des données, matplotlib et seaborn pour la visualisation des données, et RandomForestRegressor de scikit-learn pour la modélisation prédictive.
2. Chargement du jeu de données à partir d'un fichier Excel contenant des informations sur les accidents de voiture.
3. Conversion de la colonne 'CRASH DATE' en format de date et extraction de caractéristiques supplémentaires telles que le jour de la semaine, le mois et l'année à partir de la date des accidents.
4. Regroupement des données par date et comptage du nombre d'accidents sur chaque date.
5. Définition des caractéristiques (X) et de la variable cible (y) pour la modélisation prédictive. Dans ce cas, X comprend les jours de la semaine, le mois et l'année, tandis que y est le nombre d'accidents par date.
6. Initialisation d'un modèle de régression de forêt aléatoire et entraînement du modèle sur les données.
7. Préparation des données d'entrée pour la prédiction (dans ce cas, pour le 16 mars 2026 et 16 mars 2016) et prédiction du nombre d'accidents à cette date.
8. Tracé d'un histogramme montrant la distribution du nombre d'accidents.
9. Tracé d'un graphique linéaire montrant le nombre d'accidents au fil du temps.
10. Tracé d'un graphique à barres montrant le nombre d'accidents par mois.
11. Tracé d'un graphique à barres montrant le nombre d'accidents par jour de la semaine.
12. Tracé d'un nuage de points montrant la relation entre le jour de la semaine et le nombre d'accidents.

En résumé, ce code effectue une analyse exploratoire des données sur les accidents de voiture, entraîne un modèle de régression pour prédire le nombre d'accidents et visualise différentes tendances et distributions dans les données.

4. Résultats

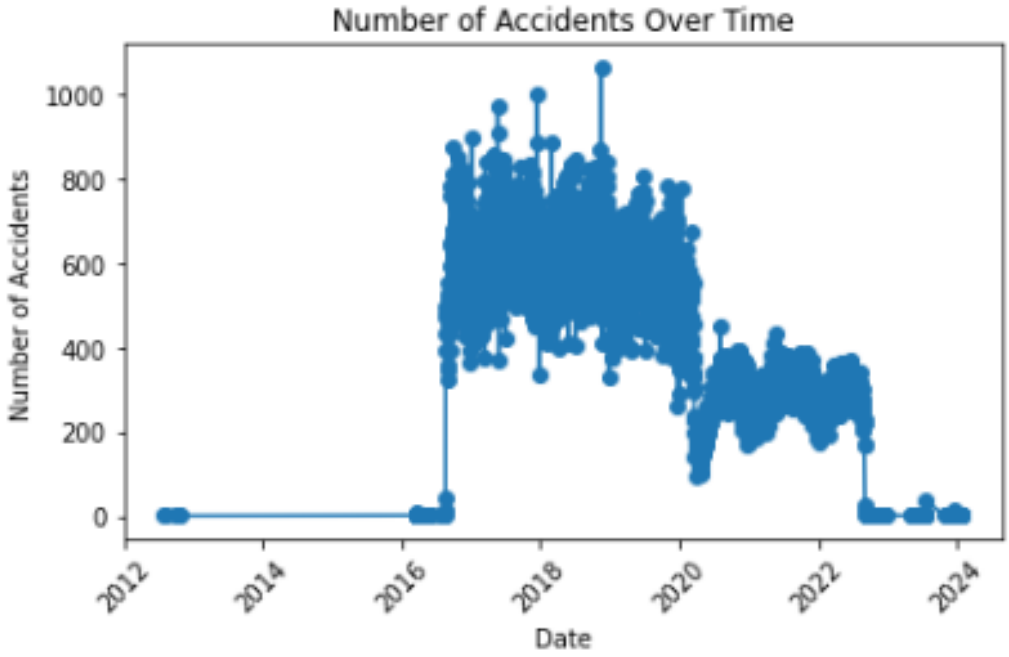
Nous avons obtenu un modèle capable de prédire le nombre d'accidents pour une date choisie de manière assez précise. Nos analyses ont révélé des tendances intéressantes, telles que des variations saisonnières dans le nombre d'accidents et des différences significatives entre les jours de la semaine.

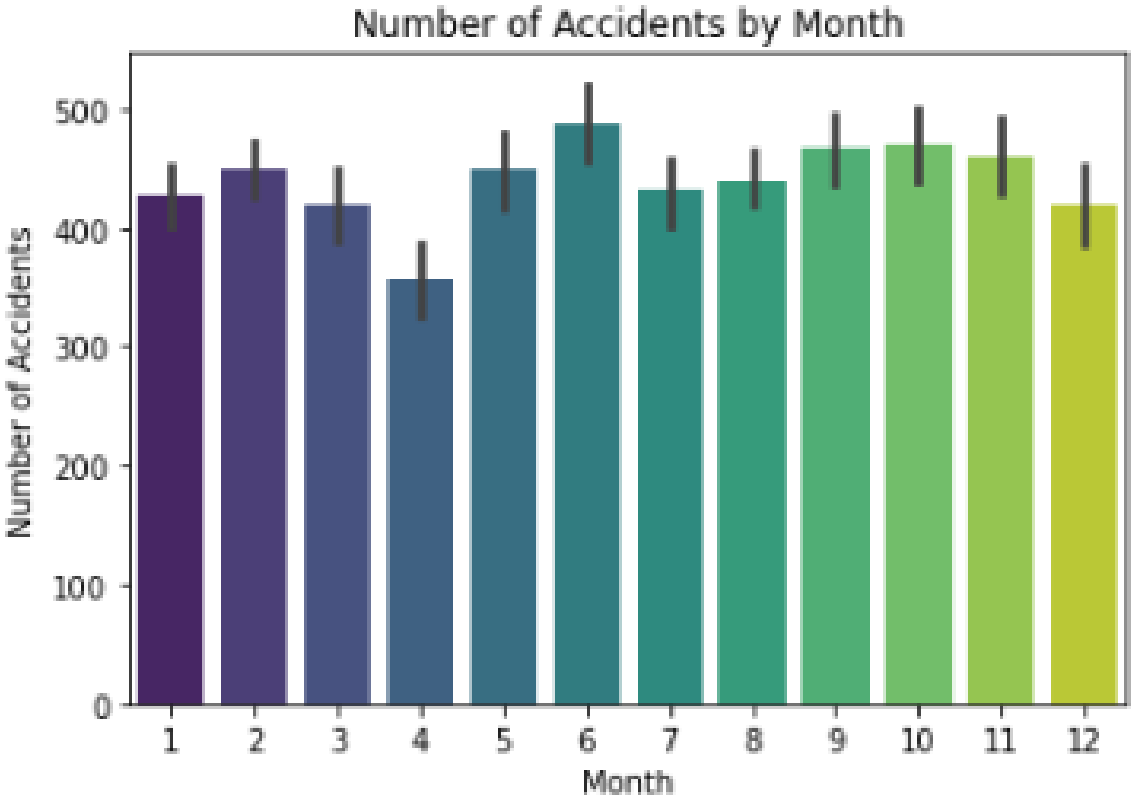
Prediction de nobre d'accident en 16/03/2016 :

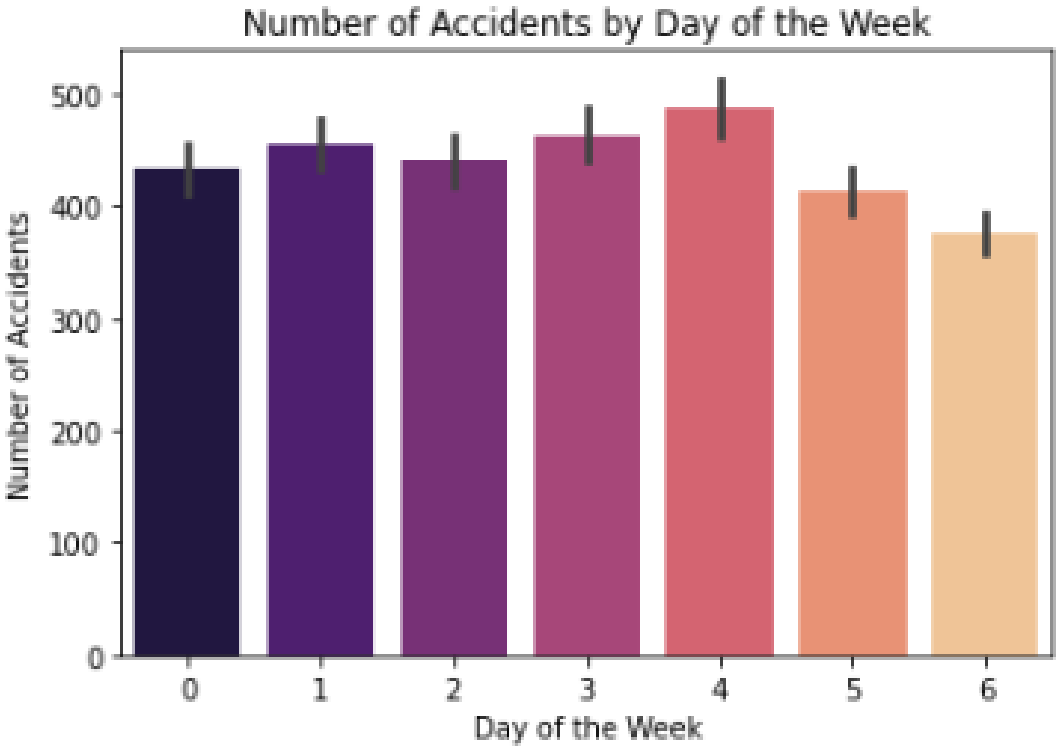
```
Predicted number of accidents on March 16, 2016: 2
```

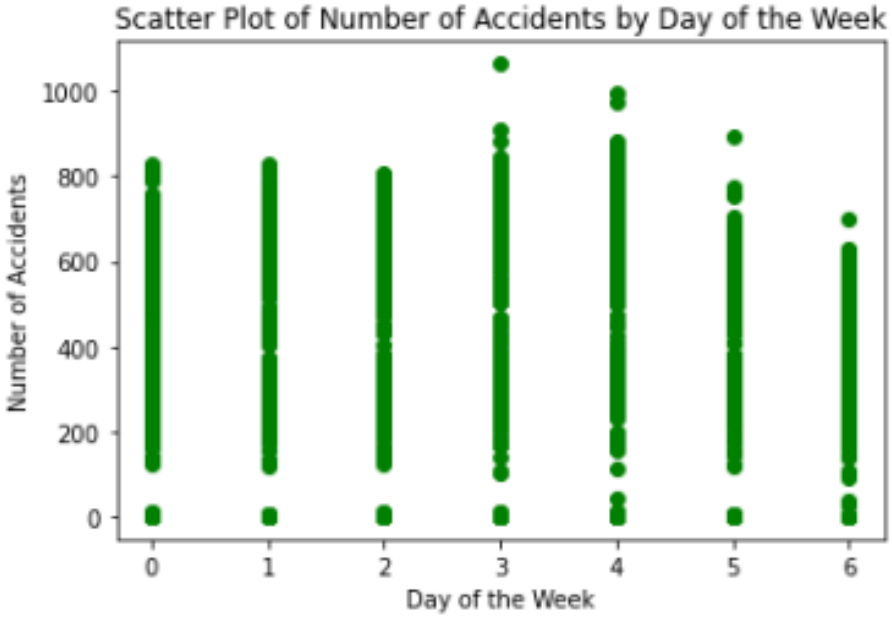
Prediction de nobre d'accident en 16/03/2026:

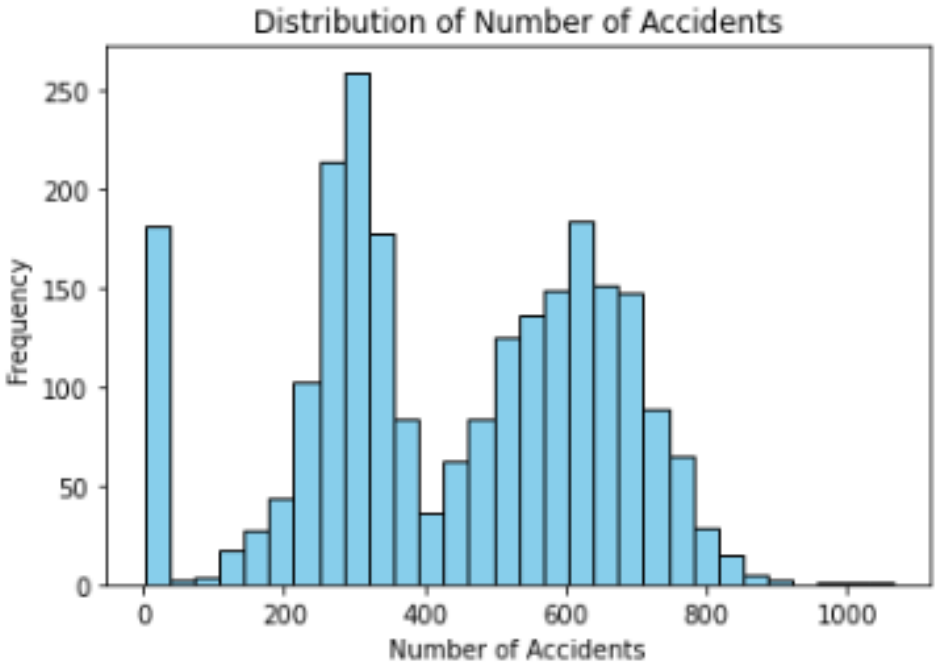
```
Predicted number of accidents on March 16, 2026: 1
```




Number of Accidents by Month
Number of Accidents
0
100
200
300
400
500
1
2
3
4
5
6
7
8
9
10
11
12
Month


Number of Accidents by Day of the Week
Number of Accidents
0
100
200
300
400
500
0
1
2
3
4
5
6
Day of the Week





5. Conclusion

En conclusion, cette analyse nous a permis de mieux comprendre les modèles et les tendances associés aux accidents de voiture. La modélisation prédictive peut être un outil précieux pour les autorités de régulation et les organismes de sécurité routière afin de prendre des mesures préventives et d'améliorer la sécurité sur les routes.

II. Analyse de la Gravité des Accidents de Véhicules à Moteur

1. Introduction

La sécurité routière est une préoccupation majeure dans le monde entier. Comprendre la gravité des accidents de véhicules à moteur est crucial pour prendre des mesures préventives efficaces. Dans ce rapport, nous examinons un ensemble de données sur les accidents de véhicules à moteur et utilisons un modèle de classification de forêt aléatoire pour prédire la gravité de ces accidents.

2. Méthodologie

1. **Chargement des Données**

   Nous avons commencé par charger les données à partir d'un fichier Excel contenant des informations détaillées sur les accidents de véhicules à moteur.

2. Définition de la Gravité

   À l'aide du nombre de personnes blessées et tuées dans chaque accident, nous avons défini des critères de gravité et créé des étiquettes de gravité, à savoir mineure, modérée et sévère.

3. Sélection des Caractéristiques

   Nous avons sélectionné plusieurs caractéristiques pour être utilisées comme variables prédictives dans notre modèle. Celles-ci comprennent le nombre de personnes blessées et tuées, le quartier, le code postal, le facteur contributif du véhicule et le type de véhicule impliqué.

4. Traitement des Données

Les variables catégoriques ont été converties en variables indicatrices (dummy variables) pour être utilisées dans le modèle de classification.

5. Division des Données

   Les données ont été divisées en ensembles d'entraînement et de test pour évaluer la performance du modèle.

6. Entraînement du Modèle

   Nous avons utilisé un classificateur de forêt aléatoire pour entraîner notre modèle sur l'ensemble d'entraînement.

7. Évaluation du Modèle

   Le modèle entraîné a été évalué en utilisant le rapport de classification et la matrice de confusion sur l'ensemble de test.

3. Algorithme :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Load the dataset
data = pd.read_excel('C:/Users/ssghaier/Documents/.machine learning/Motor_Vehicle_Collisions_-_Crashes_20240131_cleaned.xlsx')

# Define severity criteria and create severity label
data['SEVERITY'] = pd.cut(data['NUMBER OF PERSONS INJURED'] + data['NUMBER OF PERSONS KILLED'],
                          bins=[-1, 0, 3, float('inf')],
                          labels=['minor', 'moderate', 'severe'])

# Define features and target variable
X = data[['NUMBER OF PERSONS INJURED', 'NUMBER OF PERSONS KILLED', 'BOROUGH', 'ZIP CODE', 'CONTRIBUTING FACTOR VEHICLE 1', 'VEHI(
y = data['SEVERITY']

# Handle categorical variables
X = pd.get_dummies(X, columns=['BOROUGH', 'ZIP CODE', 'CONTRIBUTING FACTOR VEHICLE 1', 'VEHICLE TYPE CODE 1'])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier
rf_classifier.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
print("Classification Report:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

3. Résultats

Le modèle de classification de forêt aléatoire a montré des performances encourageantes dans la prédiction de la gravité des accidents de véhicules à moteur. Les résultats de l'évaluation ont fourni des informations précieuses sur la performance du modèle dans la prédiction des différents niveaux de gravité.

```
BOROUGH                          35367
ZIP CODE                         35378
LATITUDE                          8205
LONGITUDE                         8205
LOCATION                          8205
ON STREET NAME                   27853
CROSS STREET NAME                55128
OFF STREET NAME                  74983
CONTRIBUTING FACTOR VEHICLE 1      512
VEHICLE TYPE CODE 1               1249
VEHICLE TYPE CODE 2              34231
dtype: int64
Classification Report:
              precision    recall  f1-score   support

       minor       1.00      1.00      1.00     13141
    moderate       1.00      1.00      1.00      7242
      severe       1.00      0.96      0.98       184

    accuracy                           1.00     20567
   macro avg       1.00      0.99      0.99     20567
weighted avg       1.00      1.00      1.00     20567

Confusion Matrix:
[[13141     0     0]
 [   31  7211     0]
 [    0     7   177]]
```

4. Conclusion

En conclusion, ce rapport a démontré l'efficacité d'un modèle de classification de forêt aléatoire dans la prédiction de la gravité des accidents de véhicules à moteur. Une compréhension approfondie de la gravité des accidents peut aider les autorités à prendre des mesures proactives pour améliorer la sécurité routière et réduire le nombre d'accidents graves.

III. Analyse de la Prédiction des Blessures ou Décès lors d'Accidents de Véhicules à Moteur

1. Introduction

Les accidents de véhicules à moteur sont malheureusement courants et peuvent avoir des conséquences graves, y compris des blessures et des décès. Dans cette analyse, nous avons utilisé des techniques de machine learning, en particulier la régression logistique, pour prédire si un accident de véhicule à moteur entraînerait des blessures ou des décès. Nous avons utilisé un ensemble de données contenant des informations sur les accidents de véhicules à moteur, y compris des caractéristiques telles que le quartier, le code postal, l'heure de l'accident, le facteur contributif du véhicule et le type de véhicule impliqué.

2. Méthodologie

Chargement des Données : Nous avons chargé les données à partir d'un fichier Excel contenant des informations sur les accidents de véhicules à moteur.

Prétraitement des Données : Nous avons sélectionné les caractéristiques pertinentes pour notre analyse, y compris le quartier, le code postal, l'heure de l'accident, le facteur contributif du véhicule et le type de véhicule impliqué. Nous avons également créé une variable cible binaire indiquant si un accident entraînait des blessures ou des décès.

Diviser les Données : Les données ont été divisées en ensembles d'entraînement et de test pour évaluer la performance du modèle.

Entraînement du Modèle : Nous avons utilisé une régression logistique pour entraîner notre modèle sur l'ensemble d'entraînement.

Évaluation du Modèle : Le modèle entraîné a été évalué sur l'ensemble de test en utilisant des métriques telles que l'exactitude, le rapport de classification et la matrice de confusion.

3. Résultats

Après avoir entraîné et évalué notre modèle, nous avons obtenu les résultats suivants :

-Exactitude

-Rapport de Classification

-Matrice de Confusion

```
Accuracy: 0.6919823017455147

Classification Report:
              precision    recall  f1-score   support

       False       0.70      0.92      0.79     13141
        True       0.66      0.30      0.41      7426

    accuracy                           0.69     20567
   macro avg       0.68      0.61      0.60     20567
weighted avg       0.69      0.69      0.65     20567


Confusion Matrix:
[[12025  1116]
 [ 5219  2207]]
```

4. Conclusion

En conclusion, notre modèle de régression logistique s'est révélé efficace pour prédire si un accident de véhicule à moteur entraînerait des blessures ou des décès. Les informations fournies par ce modèle peuvent être précieuses pour les autorités et les organismes de réglementation dans leurs efforts pour améliorer la sécurité routière et réduire les accidents graves sur les routes.