Vers une Route Plus Sûre

Analyse Big Data des Accidents de la Route

Sommaire

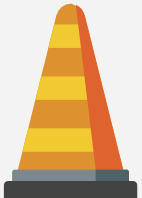
01 Introduction au projet

02 Source de données

03 Méthodologie d’extraction

04 Technologies utilisées

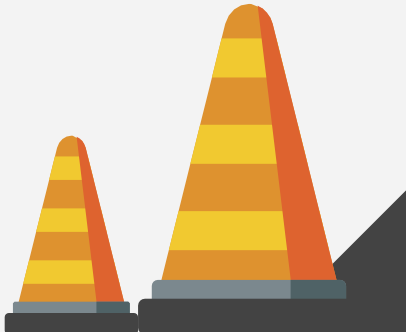
05 Aperçu des Données Extraites



Introduction

Notre projet vise à transformer une vaste collection de données sur les accidents en insights actionnables et en connaissances approfondies.

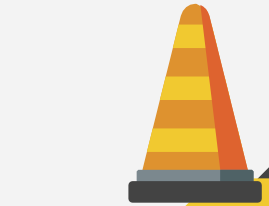
En analysant des données détaillées, notre objectif est d'identifier les causes, les tendances et les facteurs contributifs des accidents routiers, dans le but ultime de renforcer la sécurité routière et de sauver des vies.



Source de données

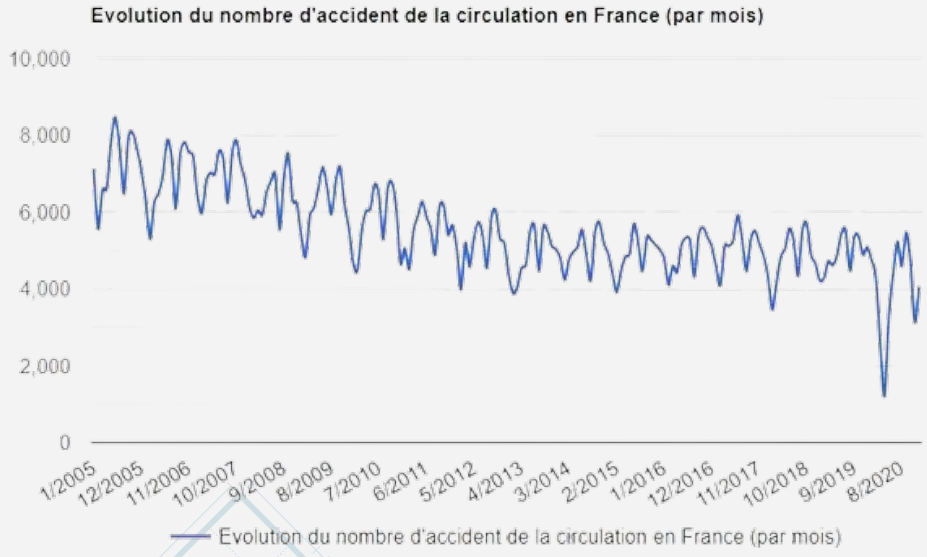
Les données que nous utilisons pour notre analyse sont extraites du site **data.gouv.fr**, qui propose un ensemble complet d'informations sur les accidents de la route en France, couvrant la période de **2005 à 2022.**

Ces ensembles de données sont organisés **annuellement en fichiers CSV**, comprenant des catégories détaillées telles que **les caractéristiques de l'accident** qui décrit les circonstances générales de l'accident **, le lieu** qui décrit le lieu principal de l'accident même si celui-ci s'est déroulé à une intersection **, les véhicules** impliqués **et les informations sur les usagers** impliqués **.**



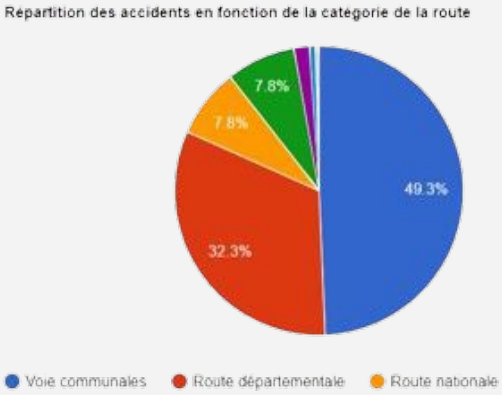
Source de données

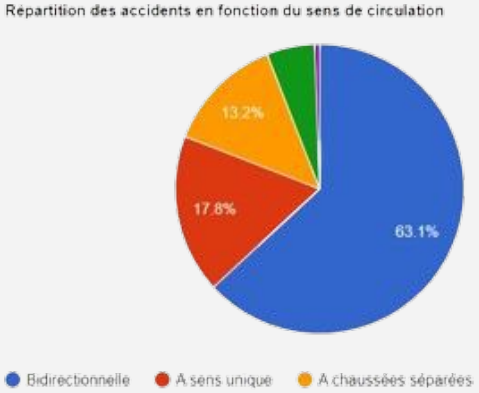
- Ce graphique linéaire montre l'évolution mensuelle du nombre d'accidents de la circulation en France sur une période étendue, s'étalant de janvier 2005 jusqu'à décembre 2022.

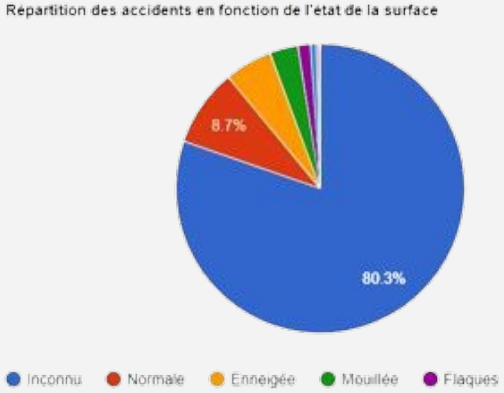


Source de données

Les trois graphiques représentent une analyse statistique des accidents de la route, montrant la répartition des accidents en fonction de la catégorie de la route, du sens de circulation et de l'état de la surface.







Méthodologie d’extraction

Notre processus d'extraction de données s'appuie sur un script Python automatisé utilisant la bibliothèque requests pour interagir avec l'API du site data.gouv.fr.

Voici les étapes clés de notre méthode :

1-Requête API : Le script initie une requête GET vers l'endpoint API spécifié, qui est conçu pour lister les ressources disponibles dans le dataset ciblé, avec un paramètre de pagination et de filtrage par mot-clé.

```python
import requests
import os

# Replace 'YOUR_API_ENDPOINT' with the actual API endpoint you want to use
api_endpoint = 'https://www.data.gouv.fr/api/2/datasets/53698f4ca3a729239d2036df/resources/?page=1&type=main&page_size=999&q='
response = requests.get(api_endpoint)
```

Méthodologie d’extraction

2 **Sélection des Données :** Nous définissons les catégories de données à télécharger, telles que 'véhicules', 'lieux', 'usagers', et 'caractéristiques', représentant les différentes dimensions des informations sur les accidents de la route.

3 **Organisation des Fichiers :** Pour chaque catégorie, le script prépare un dossier de destination dans le répertoire spécifié sur le disque local, créant un nouveau dossier si nécessaire.

4 **Téléchargement :** En parcourant la réponse JSON de l'API, le script identifie et téléchargez les fichiers pertinents à partir de leurs URL respectives, en les sauvegardant dans les dossiers correspondants.

5 **Gestion des Erreurs :** Tout au long du processus, le script vérifie les codes de statut des réponses HTTP pour s'assurer que la récupération des données et le téléchargement des fichiers se déroulent sans erreur.

Méthodologie d’extraction

```
dataToDownload = ['vehicules','lieux','usagers','caracteristiques']
basePath = 'C:/Esprit/dataScienceProject/webscrapping/'


if response.status_code == 200:
    data = response.json()
    for fileData in dataToDownload:
        download_folder = basePath+fileData+"/"
        # Create the folder if it doesn't exist
        if not os.path.exists(download_folder):
            os.makedirs(download_folder)
        # Assuming the API response contains a list of elements
        for element in data['data']:
            if 'url' in element and fileData in element['url']:
                file_url = element['url']

                # Download the file
                file_response = requests.get(file_url)

                if file_response.status_code == 200:
                    # Extract the filename from the URL
                    filename = os.path.basename(file_url)

                    # Specify the full path to save the file in the download folder
                    file_path = os.path.join(download_folder, filename)

                    # Save the file to the specified folder
                    with open(file_path, 'wb') as file:
                        file.write(file_response.content)

                    print(f"File '{filename}' downloaded successfully to '{download_folder}'.")
                else:
                    print(f"Failed to download file from {file_url}. Status code: {file_response.status_code}")
else:
    print(f"Failed to retrieve data from the API. Status code: {response.status_code}")
```

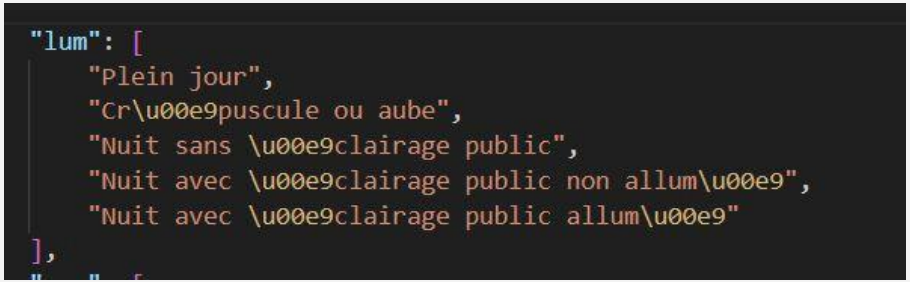
Méthodologie d’extraction

Après l'extraction des données, une étape critique a été la récupération des clés correspondant aux différentes catégories d'informations présentes dans les fichiers CSV. À partir du document descriptif fourni par l'ONISR, nous avons pu établir un fichier JSON qui contient toutes les clés et leurs valeurs possibles.

lum

Lumière : conditions d’éclairage dans lesquelles l'accident s'est produit :
1 – Plein jour
2 – Crépuscule ou aube
3 – Nuit sans éclairage public
4 – Nuit avec éclairage public non allumé
5 – Nuit avec éclairage public allumé

Description a partir du fichier description-des-bases-de-données-annuelles-2022.pdf



```
"lum": [
    "Plein jour",
    "Cr\u00e9puscule ou aube",
    "Nuit sans \u00e9clairage public",
    "Nuit avec \u00e9clairage public non allum\u00e9",
    "Nuit avec \u00e9clairage public allum\u00e9"
],
```

Clé et les valeurs possible dans le fichier keyValueResult.json

Méthodologie d’extraction

Ce script a utilisé le fichier JSON, qui agit comme une carte de référence, pour parcourir chaque CSV et remplacer les clés numériques par leurs significations textuelles correspondantes.

Nettoyage des Chaînes de Caractères : Une fonction `clean_string` est définie pour nettoyer les chaînes de caractères en éliminant les accents et caractères spéciaux grâce à la bibliothèque `unidecode`
.

```
def clean_string(s):
    """ Nettoie la chaîne de caractères en remplaçant les caractères spéciaux """
    return unidecode.unidecode(s)
```

Méthodologie d’extraction

Ce script a utilisé le fichier JSON, qui agit comme une carte de référence, pour parcourir chaque CSV et remplacer les clés numériques par leurs significations textuelles correspondantes.

Détection du Séparateur CSV : La fonction `detect_separator` lit la première ligne d'un fichier CSV pour déterminer si le séparateur des données est une virgule ou un point-virgule.

```
def detect_separator(csv_file):
    """ Détecte le séparateur dans un fichier CSV """
    with open(csv_file, 'r', encoding='ISO-8859-1') as file:
        first_line = file.readline()
        if ';' in first_line:
            return ';'
        else:
            return ','
```

Méthodologie d’extraction

Remplacement des Valeurs : La fonction `replace_values_in_csv` charge le fichier JSON de mappage, qui contient les correspondances entre les clés numériques et leurs valeurs descriptives.

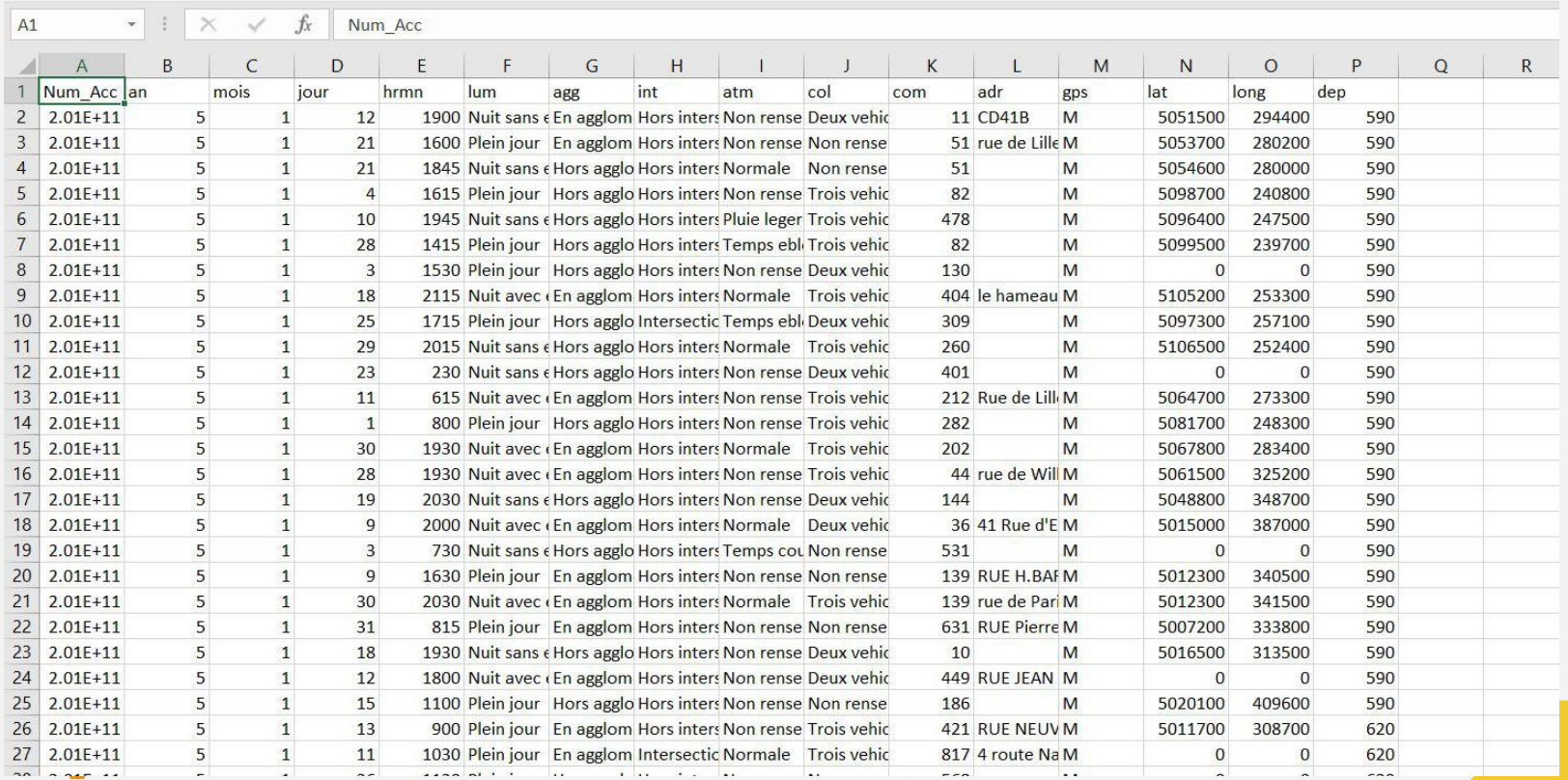
```python
def replace_values_in_csv(json_mappings_file, csv_files):
    with open(json_mappings_file, 'r', encoding='utf-8') as file:
        mappings = json.load(file)

    for key in mappings:
        if isinstance(mappings[key], list):
            mappings[key] = [clean_string(val) for val in mappings[key]]

    modified_files = []

    for csv_file in csv_files:
        try:
            sep = detect_separator(csv_file)
            df = pd.read_csv(csv_file, encoding='ISO-8859-1', sep=sep, quotechar='"')

            modified = False
            for col in df.columns:
                if col in mappings and isinstance(mappings[col], list):
                    df[col] = df[col].apply(lambda x: safe_apply(x, mappings[col]))
                    modified = True

            if modified:
                modified_csv_file = csv_file.replace('.csv', '_modified.csv')
                df.to_csv(modified_csv_file, index=False)
                modified_files.append(modified_csv_file)

        except pd.errors.ParserError as e:
            print(f"Error processing file {csv_file}: {e}")

    return modified_files
```

Technologies utilisées

Langage de programmation

Editeur de code

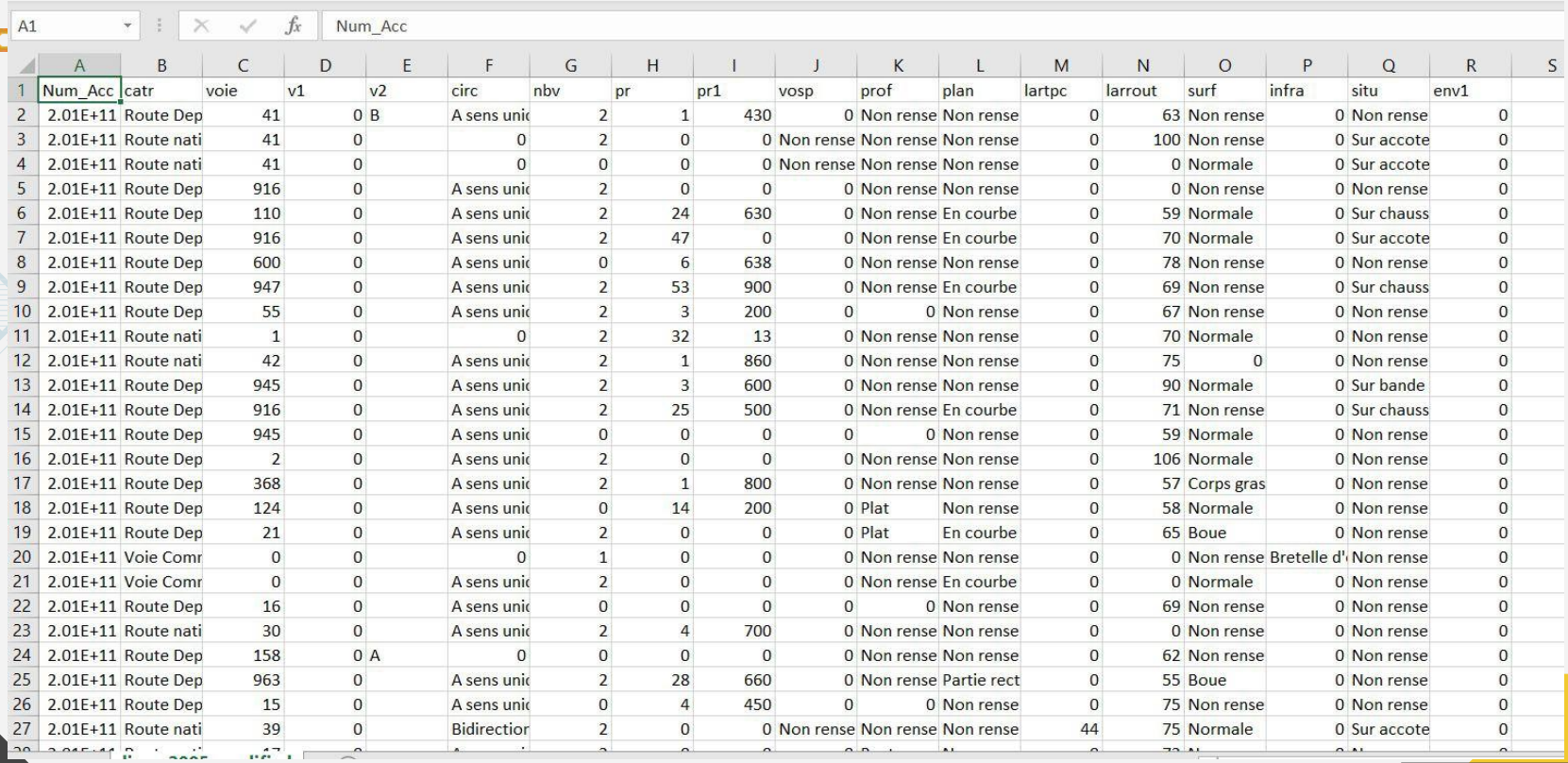
Aperçu des Données Extraites



A1 | Num_Acc

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Num_Acc	an	mois	jour	hrmn	lum	agg	int	atm	col	com	adr	gps	lat	long	dep		
2	2.01E+11	5	1	12	1900	Nuit sans é	En agglom	Hors inters	Non rense	Deux vehic	11	CD41B	M	5051500	294400	590		
3	2.01E+11	5	1	21	1600	Plein jour	En agglom	Hors inters	Non rense	Non rense	51	rue de Lille	M	5053700	280200	590		
4	2.01E+11	5	1	21	1845	Nuit sans é	Hors agglo	Hors inters	Normale	Non rense	51		M	5054600	280000	590		
5	2.01E+11	5	1	4	1615	Plein jour	Hors agglo	Hors inters	Non rense	Trois vehic	82		M	5098700	240800	590		
6	2.01E+11	5	1	10	1945	Nuit sans é	Hors agglo	Hors inters	Pluie leger	Trois vehic	478		M	5096400	247500	590		
7	2.01E+11	5	1	28	1415	Plein jour	Hors agglo	Hors inters	Temps ebl	Trois vehic	82		M	5099500	239700	590		
8	2.01E+11	5	1	3	1530	Plein jour	Hors agglo	Hors inters	Non rense	Deux vehic	130		M	0	0	590		
9	2.01E+11	5	1	18	2115	Nuit avec é	En agglom	Hors inters	Normale	Trois vehic	404	le hameau	M	5105200	253300	590		
10	2.01E+11	5	1	25	1715	Plein jour	Hors agglo	Intersectic	Temps ebl	Deux vehic	309		M	5097300	257100	590		
11	2.01E+11	5	1	29	2015	Nuit sans é	Hors agglo	Hors inters	Normale	Trois vehic	260		M	5106500	252400	590		
12	2.01E+11	5	1	23	230	Nuit sans é	Hors agglo	Hors inters	Non rense	Deux vehic	401		M	0	0	590		
13	2.01E+11	5	1	11	615	Nuit avec é	En agglom	Hors inters	Non rense	Trois vehic	212	Rue de Lill	M	5064700	273300	590		
14	2.01E+11	5	1	1	800	Plein jour	Hors agglo	Hors inters	Non rense	Trois vehic	282		M	5081700	248300	590		
15	2.01E+11	5	1	30	1930	Nuit avec é	En agglom	Hors inters	Normale	Trois vehic	202		M	5067800	283400	590		
16	2.01E+11	5	1	28	1930	Nuit avec é	En agglom	Hors inters	Non rense	Trois vehic	44	rue de Will	M	5061500	325200	590		
17	2.01E+11	5	1	19	2030	Nuit sans é	Hors agglo	Hors inters	Non rense	Deux vehic	144		M	5048800	348700	590		
18	2.01E+11	5	1	9	2000	Nuit avec é	En agglom	Hors inters	Normale	Deux vehic	36	41 Rue d'E	M	5015000	387000	590		
19	2.01E+11	5	1	3	730	Nuit sans é	Hors agglo	Hors inters	Temps cou	Non rense	531		M	0	0	590		
20	2.01E+11	5	1	9	1630	Plein jour	En agglom	Hors inters	Non rense	Non rense	139	RUE H.BAF	M	5012300	340500	590		
21	2.01E+11	5	1	30	2030	Nuit avec é	En agglom	Hors inters	Normale	Trois vehic	139	rue de Pari	M	5012300	341500	590		
22	2.01E+11	5	1	31	815	Plein jour	En agglom	Hors inters	Non rense	Non rense	631	RUE Pierre	M	5007200	333800	590		
23	2.01E+11	5	1	18	1930	Nuit sans é	Hors agglo	Hors inters	Non rense	Deux vehic	10		M	5016500	313500	590		
24	2.01E+11	5	1	12	1800	Nuit avec é	En agglom	Hors inters	Non rense	Deux vehic	449	RUE JEAN	M	0	0	590		
25	2.01E+11	5	1	15	1100	Plein jour	Hors agglo	Hors inters	Non rense	Non rense	186		M	5020100	409600	590		
26	2.01E+11	5	1	13	900	Plein jour	En agglom	Hors inters	Non rense	Trois vehic	421	RUE NEUV	M	5011700	308700	620		
27	2.01E+11	5	1	11	1030	Plein jour	En agglom	Intersectic	Normale	Trois vehic	817	4 route Na	M	0	0	620		

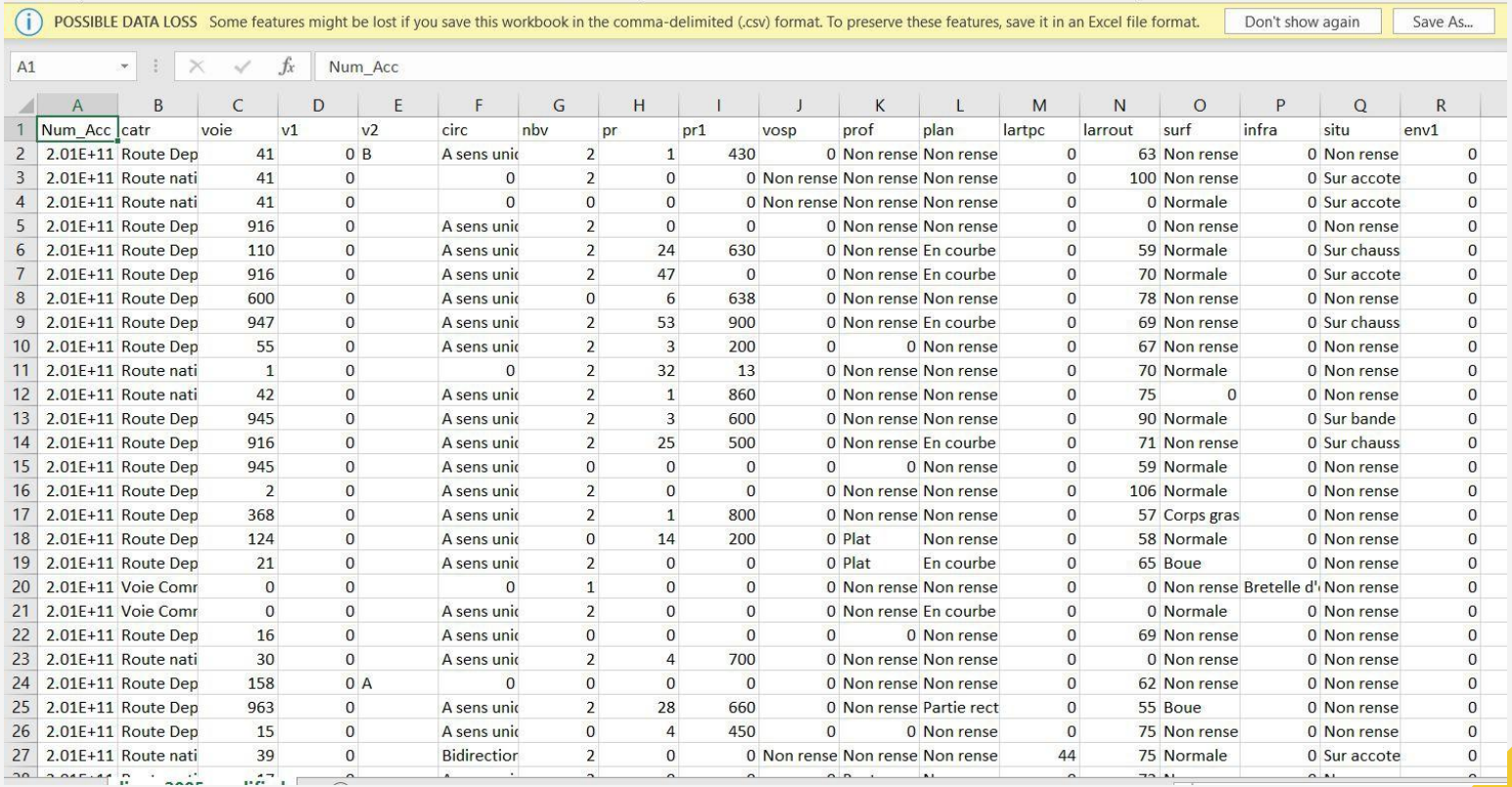
Aperçu des Données Extraites



A1 | Num_Acc

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Num_Acc	catr	voie	v1	v2	circ	nbv	pr	pr1	vosp	prof	plan	lartpc	larrout	surf	infra	situ	env1	
2	2.01E+11	Route Dep	41	0	B	A sens unic	2	1	430	0	Non rense	Non rense	0	63	Non rense	0	Non rense	0	
3	2.01E+11	Route nati	41	0		0	2	0	0	Non rense	Non rense	Non rense	0	100	Non rense	0	Sur accote	0	
4	2.01E+11	Route nati	41	0		0	0	0	0	Non rense	Non rense	Non rense	0	0	Normale	0	Sur accote	0	
5	2.01E+11	Route Dep	916	0		A sens unic	2	0	0	0	Non rense	Non rense	0	0	Non rense	0	Non rense	0	
6	2.01E+11	Route Dep	110	0		A sens unic	2	24	630	0	Non rense	En courbe	0	59	Normale	0	Sur chauss	0	
7	2.01E+11	Route Dep	916	0		A sens unic	2	47	0	0	Non rense	En courbe	0	70	Normale	0	Sur accote	0	
8	2.01E+11	Route Dep	600	0		A sens unic	0	6	638	0	Non rense	Non rense	0	78	Non rense	0	Non rense	0	
9	2.01E+11	Route Dep	947	0		A sens unic	2	53	900	0	Non rense	En courbe	0	69	Non rense	0	Sur chauss	0	
10	2.01E+11	Route Dep	55	0		A sens unic	2	3	200	0	0	Non rense	0	67	Non rense	0	Non rense	0	
11	2.01E+11	Route nati	1	0		0	2	32	13	0	Non rense	Non rense	0	70	Normale	0	Non rense	0	
12	2.01E+11	Route nati	42	0		A sens unic	2	1	860	0	Non rense	Non rense	0	75	0	0	Non rense	0	
13	2.01E+11	Route Dep	945	0		A sens unic	2	3	600	0	Non rense	Non rense	0	90	Normale	0	Sur bande	0	
14	2.01E+11	Route Dep	916	0		A sens unic	2	25	500	0	Non rense	En courbe	0	71	Non rense	0	Sur chauss	0	
15	2.01E+11	Route Dep	945	0		A sens unic	0	0	0	0	0	Non rense	0	59	Normale	0	Non rense	0	
16	2.01E+11	Route Dep	2	0		A sens unic	2	0	0	0	Non rense	Non rense	0	106	Normale	0	Non rense	0	
17	2.01E+11	Route Dep	368	0		A sens unic	2	1	800	0	Non rense	Non rense	0	57	Corps gras	0	Non rense	0	
18	2.01E+11	Route Dep	124	0		A sens unic	0	14	200	0	Plat	Non rense	0	58	Normale	0	Non rense	0	
19	2.01E+11	Route Dep	21	0		A sens unic	2	0	0	0	Plat	En courbe	0	65	Boue	0	Non rense	0	
20	2.01E+11	Voie Comr	0	0		0	1	0	0	0	Non rense	Non rense	0	0	Non rense	Bretelle d'	Non rense	0	
21	2.01E+11	Voie Comr	0	0		A sens unic	2	0	0	0	Non rense	En courbe	0	0	Normale	0	Non rense	0	
22	2.01E+11	Route Dep	16	0		A sens unic	0	0	0	0	0	Non rense	0	69	Non rense	0	Non rense	0	
23	2.01E+11	Route nati	30	0		A sens unic	2	4	700	0	Non rense	Non rense	0	0	Non rense	0	Non rense	0	
24	2.01E+11	Route Dep	158	0	A	0	0	0	0	0	Non rense	Non rense	0	62	Non rense	0	Non rense	0	
25	2.01E+11	Route Dep	963	0		A sens unic	2	28	660	0	Non rense	Partie rect	0	55	Boue	0	Non rense	0	
26	2.01E+11	Route Dep	15	0		A sens unic	0	4	450	0	0	Non rense	0	75	Non rense	0	Non rense	0	
27	2.01E+11	Route nati	39	0		Bidirectior	2	0	0	Non rense	Non rense	Non rense	44	75	Normale	0	Sur accote	0	

Aperçu des Données Extraites

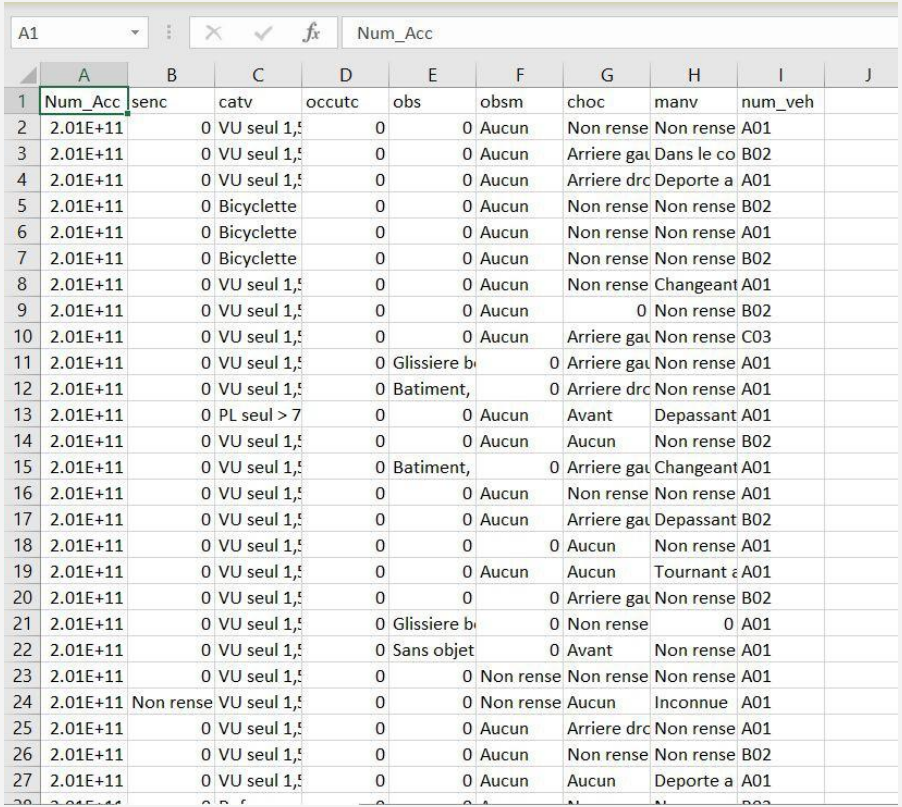


POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again Save As...

A1 Num_Acc

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Num_Acc	catr	voie	v1	v2	circ	nbv	pr	pr1	vosp	prof	plan	lartpc	larrout	surf	infra	situ	env1
2	2.01E+11	Route Dep	41	0	B	A sens unic	2	1	430	0	Non rense	Non rense	0	63	Non rense	0	Non rense	0
3	2.01E+11	Route nati	41	0		0	2	0	0	Non rense	Non rense	Non rense	0	100	Non rense	0	Sur accote	0
4	2.01E+11	Route nati	41	0		0	0	0	0	Non rense	Non rense	Non rense	0	0	Normale	0	Sur accote	0
5	2.01E+11	Route Dep	916	0		A sens unic	2	0	0	0	Non rense	Non rense	0	0	Non rense	0	Non rense	0
6	2.01E+11	Route Dep	110	0		A sens unic	2	24	630	0	Non rense	En courbe	0	59	Normale	0	Sur chauss	0
7	2.01E+11	Route Dep	916	0		A sens unic	2	47	0	0	Non rense	En courbe	0	70	Normale	0	Sur accote	0
8	2.01E+11	Route Dep	600	0		A sens unic	0	6	638	0	Non rense	Non rense	0	78	Non rense	0	Non rense	0
9	2.01E+11	Route Dep	947	0		A sens unic	2	53	900	0	Non rense	En courbe	0	69	Non rense	0	Sur chauss	0
10	2.01E+11	Route Dep	55	0		A sens unic	2	3	200	0	0	Non rense	0	67	Non rense	0	Non rense	0
11	2.01E+11	Route nati	1	0		0	2	32	13	0	Non rense	Non rense	0	70	Normale	0	Non rense	0
12	2.01E+11	Route nati	42	0		A sens unic	2	1	860	0	Non rense	Non rense	0	75	0	0	Non rense	0
13	2.01E+11	Route Dep	945	0		A sens unic	2	3	600	0	Non rense	Non rense	0	90	Normale	0	Sur bande	0
14	2.01E+11	Route Dep	916	0		A sens unic	2	25	500	0	Non rense	En courbe	0	71	Non rense	0	Sur chauss	0
15	2.01E+11	Route Dep	945	0		A sens unic	0	0	0	0	0	Non rense	0	59	Normale	0	Non rense	0
16	2.01E+11	Route Dep	2	0		A sens unic	2	0	0	0	Non rense	Non rense	0	106	Normale	0	Non rense	0
17	2.01E+11	Route Dep	368	0		A sens unic	2	1	800	0	Non rense	Non rense	0	57	Corps gras	0	Non rense	0
18	2.01E+11	Route Dep	124	0		A sens unic	0	14	200	0	Plat	Non rense	0	58	Normale	0	Non rense	0
19	2.01E+11	Route Dep	21	0		A sens unic	2	0	0	0	Plat	En courbe	0	65	Boue	0	Non rense	0
20	2.01E+11	Voie Comr	0	0		0	1	0	0	0	Non rense	Non rense	0	0	Non rense	Bretelle d'	Non rense	0
21	2.01E+11	Voie Comr	0	0		A sens unic	2	0	0	0	Non rense	En courbe	0	0	Normale	0	Non rense	0
22	2.01E+11	Route Dep	16	0		A sens unic	0	0	0	0	0	Non rense	0	69	Non rense	0	Non rense	0
23	2.01E+11	Route nati	30	0		A sens unic	2	4	700	0	Non rense	Non rense	0	0	Non rense	0	Non rense	0
24	2.01E+11	Route Dep	158	0	A	0	0	0	0	0	Non rense	Non rense	0	62	Non rense	0	Non rense	0
25	2.01E+11	Route Dep	963	0		A sens unic	2	28	660	0	Non rense	Partie rect	0	55	Boue	0	Non rense	0
26	2.01E+11	Route Dep	15	0		A sens unic	0	4	450	0	0	Non rense	0	75	Non rense	0	Non rense	0
27	2.01E+11	Route nati	39	0		Bidirectior	2	0	0	Non rense	Non rense	Non rense	44	75	Normale	0	Sur accote	0

Aperçu des Données Extraites



A1 | Num_Acc

	A	B	C	D	E	F	G	H	I	J
1	Num_Acc	senc	catv	occutc	obs	obsm	choc	manv	num_veh	
2	2.01E+11	0	VU seul 1,5	0	0	Aucun	Non rense	Non rense	A01	
3	2.01E+11	0	VU seul 1,5	0	0	Aucun	Arriere gau	Dans le co	B02	
4	2.01E+11	0	VU seul 1,5	0	0	Aucun	Arriere drc	Deporte a	A01	
5	2.01E+11	0	Bicyclette	0	0	Aucun	Non rense	Non rense	B02	
6	2.01E+11	0	Bicyclette	0	0	Aucun	Non rense	Non rense	A01	
7	2.01E+11	0	Bicyclette	0	0	Aucun	Non rense	Non rense	B02	
8	2.01E+11	0	VU seul 1,5	0	0	Aucun	Non rense	Changeant	A01	
9	2.01E+11	0	VU seul 1,5	0	0	Aucun	0	Non rense	B02	
10	2.01E+11	0	VU seul 1,5	0	0	Aucun	Arriere gau	Non rense	C03	
11	2.01E+11	0	VU seul 1,5	0	Glissiere b	0	Arriere gau	Non rense	A01	
12	2.01E+11	0	VU seul 1,5	0	Batiment,	0	Arriere drc	Non rense	A01	
13	2.01E+11	0	PL seul > 7	0	0	Aucun	Avant	Depassant	A01	
14	2.01E+11	0	VU seul 1,5	0	0	Aucun	Aucun	Non rense	B02	
15	2.01E+11	0	VU seul 1,5	0	Batiment,	0	Arriere gau	Changeant	A01	
16	2.01E+11	0	VU seul 1,5	0	0	Aucun	Non rense	Non rense	A01	
17	2.01E+11	0	VU seul 1,5	0	0	Aucun	Arriere gau	Depassant	B02	
18	2.01E+11	0	VU seul 1,5	0	0	0	Aucun	Non rense	A01	
19	2.01E+11	0	VU seul 1,5	0	0	Aucun	Aucun	Tournant à	A01	
20	2.01E+11	0	VU seul 1,5	0	0	0	Arriere gau	Non rense	B02	
21	2.01E+11	0	VU seul 1,5	0	Glissiere b	0	Non rense	0	A01	
22	2.01E+11	0	VU seul 1,5	0	Sans objet	0	Avant	Non rense	A01	
23	2.01E+11	0	VU seul 1,5	0	0	Non rense	Non rense	Non rense	A01	
24	2.01E+11	Non rense	VU seul 1,5	0	0	Non rense	Aucun	Inconnue	A01	
25	2.01E+11	0	VU seul 1,5	0	0	Aucun	Arriere drc	Non rense	A01	
26	2.01E+11	0	VU seul 1,5	0	0	Aucun	Non rense	Non rense	B02	
27	2.01E+11	0	VU seul 1,5	0	0	Aucun	Aucun	Deporte a	A01	

Préparation des données

- Nettoyage des données à travers des scripts python
- Extraction des données de la météo et création de la dimension temps
- ETL en utilisant SSIS et insertion dans la base

Langage de programmation

Logiciel de migration de données

Formatage des dates

caracteristiques.py 3

caracteristiques.py > process_csv

```python
    # Preprocessing for 'hrmn' to ensure it's in a proper 4-digit format
    df["hrmn"] = df["hrmn"].astype(str).str.zfill(4)

    df["mois"] = df["mois"].astype(str).str.zfill(2)
    df["jour"] = df["jour"].astype(str).str.zfill(2)
    df["hour"] = df["hrmn"].str[:2]  # Extract hour
    df["minute"] = df["hrmn"].str[-2:]  # Extract minute


    df['an'] = df['an'].apply(lambda x: (2000 + int(x)) if len(str(x)) != 4 else int(x))
    df['an'] = df['an'].astype(str)

    # Create 'Date' column by combining the year, month, day, hour, and minute
    df["Date"] = pd.to_datetime(
        df["an"]
        + "-"
        + df["mois"]
        + "-"
        + df["jour"]
        + " "
        + df["hour"]
        + ":"
        + df["minute"]
    )

    # Replace empty values with 0 in 'lat' and 'long', ensure numeric type
    df["lat"] = pd.to_numeric(df["lat"], errors="coerce").fillna(0)
    # Convert latitude to degrees
    df["lat"] = df["lat"] / 111319.9
    df["long"] = pd.to_numeric(df["long"], errors="coerce").fillna(0)
    df["long"] = df.apply(
        lambda row: row["long"]
        / (111319.9 * abs(np.cos(np.radians(row["lat"] / 1000000)))),
        axis=1,
    )

    # Replace empty values with 'M' in 'gps' if it exists else, create a column  with 'M' as value
    df['gps'] = 'M' if 'gps' not in df.columns else df["gps"].fillna("M")

    # Remove the specified columns
    columns_to_drop = ["an", "mois", "jour", "hrmn", "hour", "minute"]
    df.drop(columns=columns_to_drop, inplace=True)
```

usagers.py 3

usagers.py > { } pd

```python
import pandas as pd
import numpy as np
import os
import openpyxl


# Function to process each CSV file
def process_csv(file_path):
    # Load the CSV file
    df = pd.read_csv(file_path)

    # format year
    df["an_nais"] = df["an_nais"].fillna(0)
    df["an_nais"] = df["an_nais"].apply(
        lambda x: int(x)
    )
    df["an_nais"] = df["an_nais"].astype(str)

    # Extract base name (without extension) to use as the output file name
    file_name = os.path.splitext(os.path.basename(file_path))[0]
    file_name = file_name.replace(file_name.split("_")[2], "final")

    # Specify the output XLSX file with the modified file_name
    xlsx_file = os.path.join(
        output_dir, f"{file_name}.xlsx"
    )  # Output file will have the modified file_name

    # Save to xlsx
    df.to_excel(
        xlsx_file, index=False, engine="openpyxl"
    )  # Export DataFrame to XLSX without the index

    print(f"File {file_path} has been processed and saved as {xlsx_file}")


# Directory paths
input_dir = "usagers"
output_dir = "usagers_final"

# Create output directory if not exists
os.makedirs(output_dir, exist_ok=True)

```

Problèmes de mise en forme des valeurs manquantes

lieux.py 3 ×

lieux.py > ...

```python
# Function to process each CSV file
def process_csv(file_path):
    # Load the CSV file
    df = pd.read_csv(file_path)

    # fill v2 missing values with N/A
    df["v2"] = df["v2"].fillna("N/A")

    # Extract base name (without extension) to use as the output file name
    file_name = os.path.splitext(os.path.basename(file_path))[0]
    file_name = file_name.replace(file_name.split("_")[2], "final")

    # Specify the output XLSX file with the modified file_name
    xlsx_file = os.path.join(
        output_dir, f"{file_name}.xlsx"
    )  # Output file will have the modified file_name

    # Save to xlsx
    df.to_excel(
        xlsx_file, index=False, engine="openpyxl"
    )  # Export DataFrame to XLSX without the index

    print(f"File {file_path} has been processed and saved as {xlsx_file}")


# Directory paths
input_dir = "lieux"
output_dir = "lieux_final"

# Create output directory if not exists
os.makedirs(output_dir, exist_ok=True)

# Process each CSV file in the input directory
for file_name in os.listdir(input_dir):
    if file_name.endswith(".csv"):
        file_path = os.path.join(input_dir, file_name)
        process_csv(file_path)

print("All files have been processed.")

```

vehicules.py 3 ×

vehicules.py > { } pd

```python
import pandas as pd
import numpy as np
import os
import openpyxl

# Function to process each CSV file
def process_csv(file_path):
    # Load the CSV file
    df = pd.read_csv(file_path)

    # Extract base name (without extension) to use as the output file name
    file_name = os.path.splitext(os.path.basename(file_path))[0]
    file_name = file_name.replace(file_name.split("_")[2], "final")

    # Specify the output XLSX file with the modified file_name
    xlsx_file = os.path.join(
        output_dir, f"{file_name}.xlsx"
    )  # Output file will have the modified file_name

    # Save to xlsx
    df.to_excel(
        xlsx_file, index=False, engine="openpyxl"
    )  # Export DataFrame to XLSX without the index

    print(f"File {file_path} has been processed and saved as {xlsx_file}")


# Directory paths
input_dir = "vehicules"
output_dir = "vehicules_final"

# Create output directory if not exists
os.makedirs(output_dir, exist_ok=True)

# Process each CSV file in the input directory
for file_name in os.listdir(input_dir):
    if file_name.endswith(".csv"):
        file_path = os.path.join(input_dir, file_name)
        process_csv(file_path)

print("All files have been processed.")

```

Mappage des valeurs numériques avec leur labels

Script.py 2

Script.py > {} pd

```python
import pandas as pd
import os
import numpy as np

# Define your directories
input_directory = 'lieux_final'  # Update this with your input directory path
output_directory = 'lieux_finalModified'  # Update this with your output directory path

# Ensure the output directory exists
if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# Mapping definitions for each column
column_mappings = {
    'catr': {
        '0' : 'autre',
        '1': 'Autoroute',
        '2': 'Route nationale',
        '3': 'Route Départementale',
        '4': 'Voie Communales',
        '5': 'Hors réseau public',
        '6': 'Parc de stationnement ouvert à la circulation publique',
        '7': 'Routes de métropole urbaine',
        '9': 'autre',
        '9.0': 'autre',
        '-1': 'Non renseigne',  # Assuming -1 is used for not provided
    },
    'circ': {
        '1': 'A sens unique',
        '0' : 'autre',
        '0.0' : 'autre',
        '2': 'Bidirectionnelle',
        '3': 'A chaussées séparées',
        '4': 'Avec voies d'affectation variable',
        '-1': 'Non renseigne',
    },
    'vosp': {
        '0': 'Sans objet',
        '0.0': 'Sans objet',
        '1': 'Piste cyclable',
        '2': 'Bande cyclable',
        '3': 'Voie réservée',
```

Script.py 2

Script.py > {} pd

```python
        '8' :'Chantier',
        '9' :'Autres',
    },
    'situ' : {
        '-1' : 'Non renseigne',
        '0' :'Aucun',
        '0.0' : 'Aucun',
        '1' :'Sur chaussée',
        '2' :'Sur bande d'arrêt d'urgence',
        '3' :'Sur accotement',
        '4' :'Sur trottoir',
        '5' :'Sur piste cyclable',
        '6' :'Sur autre voie spéciale',
        '8' :'Autres',
    },
}

def apply_mappings(df, mappings):
    for column, mapping in mappings.items():
        # Check if the column exists to avoid KeyError
        if column in df:
            # Apply mapping, use df[column].map(mapping) if all values are expected to be in the mapping
            df[column] = df[column].apply(lambda x: mapping.get(str(x), x))
    return df

# Process each file in the input directory
for filename in os.listdir(input_directory):
    if filename.endswith(".xlsx") and 'final' in filename:
        file_path = os.path.join(input_directory, filename)
        df = pd.read_excel(file_path)

        # Apply mappings
        df = apply_mappings(df, column_mappings)

        # Construct the new filename and path for output
        new_filename = filename.replace('final', 'finalModified')
        output_path = os.path.join(output_directory, new_filename)

        # Save the updated DataFrame
        df.to_excel(output_path, index=False, engine='openpyxl')

        print(f"Processed and saved {new_filename} in {output_directory}")
```

Mappage des valeurs numériques avec leur labels

Secu.py 1 ×

Secu.py > {} pd

```python
import pandas as pd
import os

# Define your directories
input_directory = 'usagers_final'  # Update this with your input directory path
output_directory = 'usagers_finalModified'  # Update this with your output directory path

# Ensure the output directory exists
if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# Mapping definitions for each column
mapping = {
    '-1': 'Non renseigne',
    '0': 'Aucun équipement',
    '0': 'Non renseigne',
    '1': 'Ceinture',
    '2': 'Casque',
    '3': 'Dispositif enfants',
    '10': 'Ceinture',
    '11': 'Ceinture',
    '12': 'Ceinture',
    '13': 'Ceinture',
    '20': 'Casque',
    '21': 'Casque',
    '22': 'Casque',
    '23': 'Casque',
    '30': 'Dispositif enfants',
    '31': 'Dispositif enfants',
    '32': 'Dispositif enfants',
    '33': 'Dispositif enfants',
    '40': 'Gilet réfléchissant',
    '41': 'Gilet réfléchissant',
    '42': 'Gilet réfléchissant',
    '43': 'Gilet réfléchissant',
    '90': 'Autre',
    '91': 'Autre',
    '92': 'Autre',
    '93': 'Autre',
    '5': 'Airbag (2RM/3RM)',
    '6': 'Gants (2RM/3RM)',
    '7': 'Gants + Airbag (2RM/3RM)',
```

Secu.py 1 ×

Secu.py > {} pd

```python
    '41': 'Gilet réfléchissant',
    '42': 'Gilet réfléchissant',
    '43': 'Gilet réfléchissant',
    '90': 'Autre',
    '91': 'Autre',
    '92': 'Autre',
    '93': 'Autre',
    '5': 'Airbag (2RM/3RM)',
    '6': 'Gants (2RM/3RM)',
    '7': 'Gants + Airbag (2RM/3RM)',
    '8': 'Non déterminable',
    '9': 'Autre',
}

def map_values(key):
    return mapping.get(key, key)

# Process each file in the input directory
for filename in os.listdir(input_directory):
    if filename.endswith(".xlsx") and 'final' in filename:
        file_path = os.path.join(input_directory, filename)
        df = pd.read_excel(file_path,dtype={'secu':str})

        # Apply mappings
        print("Before mapping:")
        print(df['secu'])

        df['secu'] = df['secu'].astype(str)
        df['secu'] = df['secu'].apply(map_values)

        print("After mapping:")
        print(df['secu'])

        # Construct the new filename and path for output
        new_filename = filename.replace('final', 'finalModified')
        output_path = os.path.join(output_directory, new_filename)

        # Save the updated DataFrame
        df.to_excel(output_path, index=False, engine='openpyxl')

        print(f"Processed and saved {new_filename} in {output_directory}")
```

Mappage des valeurs numériques avec leur labels

Script_Caracteristique.py 2

Script_Caracteristique.py > {} pd

```python
import pandas as pd
import os
import numpy as np

# Define your directories
input_directory = 'caracteristiques_final'  # Update this with your input directory path
output_directory = 'caracteristiques_finalModified'  # Update this with your output directory path

# Ensure the output directory exists
if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# Mapping definitions for each column
column_mappings = {
    'lum': {
        '-1' : 'Non renseigne',
        '0' : 'autre',
        '1': 'Plein jour',
        '2': 'Crépuscule ou aube',
        '3': 'Nuit sans éclairage public',
        '4': 'Nuit avec éclairage public non allumé',
        '5': 'Nuit avec éclairage public allumé',
    },
    'int': {
        '-1' : 'Non renseigne',
        '1': 'Hors intersection',
        '0' : 'autre',
        '2': 'Intersection en X',
        '3': 'Intersection en T',
        '4': 'Intersection en Y',
        '5': 'Intersection à plus de 4 branches',
        '6': 'Giratoire',
        '7': 'Place',
        '8': 'Passage à niveau',
        '9': 'Autre intersection',
    },
    'atm': {
        '-1' : 'Non renseigne',
        '0': 'Sans objet',
        '1': 'Normale',
        '2': 'Pluie légère',
        '3': 'Pluie forte',
    },
    'col' : {
        '-1':  'Non renseigne',
        '0' : 'Non renseigne',
        '0.0' : 'Non renseigne',
        '1' : 'Deux véhicules - frontale',
        '2' : 'Deux véhicules - par l'arrière ',
        '3' : 'Deux véhicules - par le coté',
        '4' : 'Trois véhicules et plus - en chaîne',
        '5' : 'Trois véhicules et plus - collisions multiples',
        '6' : 'Autre collision',
        '7' : 'Sans collision',
        '9' : 'Autre',
    },
}

def apply_mappings(df, mappings):
    for column, mapping in mappings.items():
        # Check if the column exists to avoid KeyError
        if column in df:
            # Apply mapping, use df[column].map(mapping) if all values are expected to be in the mapping
            df[column] = df[column].apply(lambda x: mapping.get(str(x), x))
    return df

# Process each file in the input directory
for filename in os.listdir(input_directory):
    if filename.endswith(".xlsx") and 'final' in filename:
        file_path = os.path.join(input_directory, filename)
        df = pd.read_excel(file_path)

        # Apply mappings
        df = apply_mappings(df, column_mappings)

        # Construct the new filename and path for output
        new_filename = filename.replace('final', 'finalModified')
        output_path = os.path.join(output_directory, new_filename)

        # Save the updated DataFrame
        df.to_excel(output_path, index=False, engine='openpyxl')

        print(f"Processed and saved {new_filename} in {output_directory}")
```

Mappage des valeurs numériques avec leur labels

Script_Usager.py 2

Script_Usager.py > {} pd

```python
import pandas as pd
import os
import numpy as np

# Define your directories
input_directory = 'usagers_final'  # Update this with your input directory path
output_directory = 'usagers_finalModified'  # Update this with your output directory path

# Ensure the output directory exists
if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# Mapping definitions for each column
column_mappings = {
    'catu': {
        '0' : 'autre',
        '1': 'Conducteur',
        '2': 'Passager',
        '3': 'Piéton',
        '4': 'Autre',
    },
    'grav': {
        '-1' : 'Non renseigne',
        '0' : 'autre',
        '1': 'Indemne',
        '2': 'Tué',
        '3': 'Blessé hospitalisé',
        '4': 'Blessé léger',
    },
    'sexe': {
        '-1' : 'Non renseigne',
        '0': 'Sans objet',
        '1': 'Masculin',
        '2': 'Féminin',
    },
    'trajet' : {
        '-1' : 'Non renseigne',
        '0' : 'Non renseigne',
        '0.0' : 'Non renseigne',
        '1' : 'Domicile - travail',
        '2' : 'Domicile - école',
        '3' : 'Courses - achats',
        '4' : 'Masqué',
        '5' : 'Jouant - courant ',
        '6' : 'Avec animal',
        'A' : 'Monte/descend du véhicule',
        'B' : 'Inconnue',
        '9' : 'Autre',
    },
    'etatp' : {
        '-1':  'Non renseigne',
        '0' : 'Non renseigné ou sans objet',
        '0.0' : 'Non renseigné ou sans objet',
        '1' : 'Seul',
        '2' : 'Accompagné',
        '3' : 'En groupe',
    },
}

def apply_mappings(df, mappings):
    for column, mapping in mappings.items():
        # Check if the column exists to avoid KeyError
        if column in df:
            # Apply mapping, use df[column].map(mapping) if all values are expected to be in the mapping
            df[column] = df[column].apply(lambda x: mapping.get(str(x), x))
    return df

# Process each file in the input directory
for filename in os.listdir(input_directory):
    if filename.endswith(".xlsx") and 'final' in filename:
        file_path = os.path.join(input_directory, filename)
        df = pd.read_excel(file_path,dtype={'secu':str})

        # Apply mappings
        df = apply_mappings(df, column_mappings)

        # Construct the new filename and path for output
        new_filename = filename.replace('final', 'finalModified')
        output_path = os.path.join(output_directory, new_filename)

        # Save the updated DataFrame
        df.to_excel(output_path, index=False, engine='openpyxl')

        print(f"Processed and saved {new_filename} in {output_directory}")
```

Mappage des valeurs numériques avec leur labels

Script_vehicule.py 2

```python
import pandas as pd
import os
import numpy as np

# Define your directories
input_directory = 'vehicules_final'  # Update this with your input directory path
output_directory = 'vehicules_finalModified'  # Update this with your output directory path

# Ensure the output directory exists
if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# Mapping definitions for each column
column_mappings = {
    'senc': {
        '-1' : 'Non renseigne',
        '0' : 'Inconnu',
        '0.0' : 'Inconnu',
        '1': 'PK ou PR ou numéro d'adresse postale croissant',
        '2': 'PK ou PR ou numéro d'adresse postale décroissant',
        '3': 'Absence de repère',
    },
    'catv' : {
        '-1' : 'Non renseigne',
        '0' : 'Non renseigne',
        '00': 'Indéterminable',
        '01': 'Bicyclette',
        '02': 'Cyclomoteur <50cm3',
        '03': 'Voiturette (Quadricycle à moteur carrossé)',
        '04': 'Référence inutilisée depuis 2006 (scooter immatriculé)',
        '05': 'Référence inutilisée depuis 2006 (motocyclette)',
        '06': 'Référence inutilisée depuis 2006 (side-car)',
        '07': 'VL seul',
        '08': 'Référence inutilisée depuis 2006 (VL + caravane)',
        '09': 'Référence inutilisée depuis 2006 (VL + remorque)',
        '10': 'VU seul 1,5T <= PTAC <= 3,5T avec ou sans remorque',
        '11': 'Référence inutilisée depuis 2006 (VU + caravane)',
        '12': 'Référence inutilisée depuis 2006 (VU + remorque)',
        '13': 'PL seul 3,5T <PTCA <= 7,5T',
        '14': 'PL seul > 7,5T',
        '15': 'PL > 3,5T + remorque',
        '16': 'Tracteur routier seul',
```

Script_vehicule.py 2

```python
        '26': 'Autres manœuvres',
    },
    'motor' : {
        '-1' : 'Non renseigne',
        '0' : 'Inconnue',
        '0.0' : 'Inconnue',
        '1' : 'Hydrocarbures',
        '2':  'Hybride électrique',
        '3' : 'Electrique',
        '4' : 'Hydrogène',
        '5' : 'Humaine',
        '6' :'Autre',
    },
}

def apply_mappings(df, mappings):
    for column, mapping in mappings.items():
        # Check if the column exists to avoid KeyError
        if column in df:
            # Apply mapping, use df[column].map(mapping) if all values are expected to be in the mapping
            df[column] = df[column].apply(lambda x: mapping.get(str(x), x))
    return df

# Process each file in the input directory
for filename in os.listdir(input_directory):
    if filename.endswith(".xlsx") and 'final' in filename:
        file_path = os.path.join(input_directory, filename)
        df = pd.read_excel(file_path)

        # Apply mappings
        df = apply_mappings(df, column_mappings)

        # Construct the new filename and path for output
        new_filename = filename.replace('final', 'finalModified')
        output_path = os.path.join(output_directory, new_filename)

        # Save the updated DataFrame
        df.to_excel(output_path, index=False, engine='openpyxl')

        print(f"Processed and saved {new_filename} in {output_directory}")
```

Création de la dimension temps

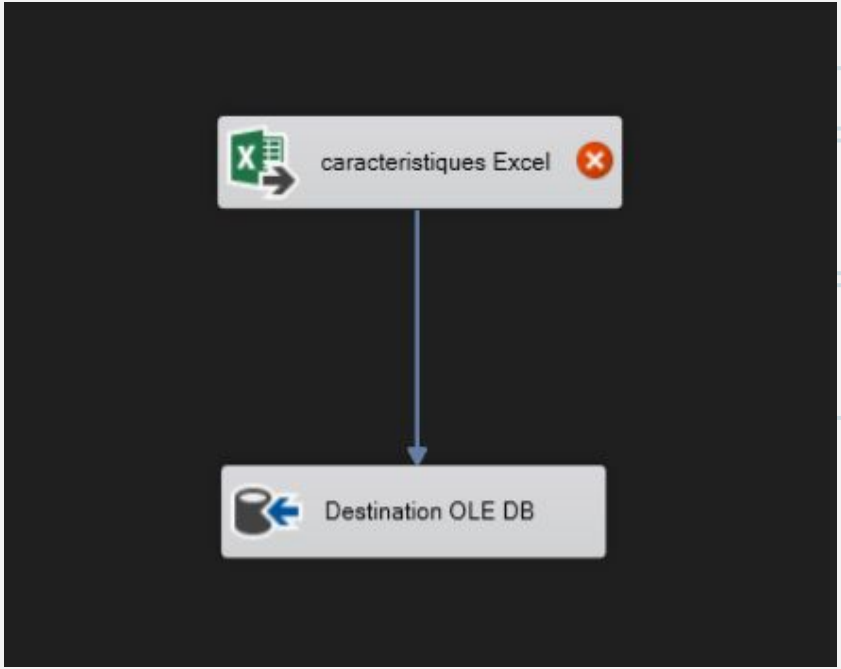
```
WITH DateCTE
AS (
    SELECT cast('20050101' AS DATETIME) Datevalue

    UNION ALL

    SELECT datevalue + 1
    FROM DateCTE
    WHERE datevalue + 1 < = '20251231'
    )
        SELECT CAST(CONVERT(VARCHAR(8), Datevalue, 112) AS INT) AS ID_Temps
    ,CAST(Datevalue AS DATE) AS [Date]
    ,cast(DATENAME(d, datevalue) AS NVARCHAR(10))+' - '+cast(DATENAME(Month, DateValue) AS NVARCHAR(30))+' - '+cast(DATENAME(year, DateValue) AS NVARCHAR(30)) AS [Jour_Mois_Annee
    ,cast(DATENAME(year, DateValue) AS INT) AS [Annee]
    ,cast(DATENAME(year, DateValue) AS NVARCHAR(4))+' Semestre '+ cast(ROUND ( cast (DATEPART(quarter, CAST(Datevalue AS DATE)) as float)/cast (2 as float),0) AS NVARCHAR(11)) as
    ,'Semestre '+ cast(ROUND ( cast (DATEPART(quarter, CAST(Datevalue AS DATE)) as float)/cast (2 as float),0) AS NVARCHAR(11)) as [Semestre]
    ,cast(DATENAME(year, DateValue) AS NVARCHAR(4))+' Trimestre '+ cast(DATEPART (quarter, CAST(Datevalue AS DATE)) AS NVARCHAR(11)) as [ID_Trimestre]
    ,'Trimestre '+ cast(DATEPART (quarter, CAST(Datevalue AS DATE)) AS NVARCHAR(11)) as [Trimestre]
    ,cast(DATENAME(year, DateValue) + REPLICATE('0', 2 - LEN(Month(DateValue))) + CAST(Month(DateValue) AS VARCHAR) AS INT) AS ID_Mois
    ,cast(MONTH(DateValue) AS INT) AS [Mois]
    ,cast(DATENAME(Month, DateValue) AS NVARCHAR(30)) AS [Lib_Mois]
    ,cast(DATENAME(d, datevalue) AS INT) AS [Jour]
    ,DATEPART(DW, datevalue) AS [Id_Lib_Jour]
    ,cast(DATENAME(DW, Datevalue) AS NVARCHAR(10)) AS [Lib_Jour]
    ,cast(DATENAME(WEEK, Datevalue) AS INT) AS [Semaine]
    ,cast(DATENAME(dayofyear, Datevalue) AS INT) AS [JourDeAnnee]
    ,cast(DATENAME(DW, Datevalue) AS NVARCHAR(10))+' - '+cast(DATENAME(d, datevalue) AS NVARCHAR(10))+' - '+cast(DATENAME(Month, DateValue)AS NVARCHAR(30))  AS [Jour_mois_lettre]
    ,case
        when cast(DATENAME(d, datevalue) AS INT) between 1 and 10 then (DATENAME(year, DateValue) + REPLICATE('0', 2 - LEN(Month(DateValue))) + CAST(Month(DateValue)  AS VARCHAR)+
        when cast(DATENAME(d, datevalue) AS INT) between 11 and 20 then (DATENAME(year, DateValue) + REPLICATE('0', 2 - LEN(Month(DateValue))) + CAST(Month(DateValue)  AS VARCHAR
        else (DATENAME(year, DateValue) + REPLICATE('0', 2 - LEN(Month(DateValue))) + CAST(Month(DateValue)  AS VARCHAR)+'D3')
    END AS [ID_Decade]
    ,case
        when cast(DATENAME(d, datevalue) AS INT) between 1 and 10 then 'D1'
        when cast(DATENAME(d, datevalue) AS INT) between 11 and 20 then 'D2'
        else 'D3'
    END AS [Decade]

into DIM_TEMPS FROM DateCTE D

ORDER BY Datevalue
OPTION (MAXRECURSION 0)
```

Extraction des données de météo

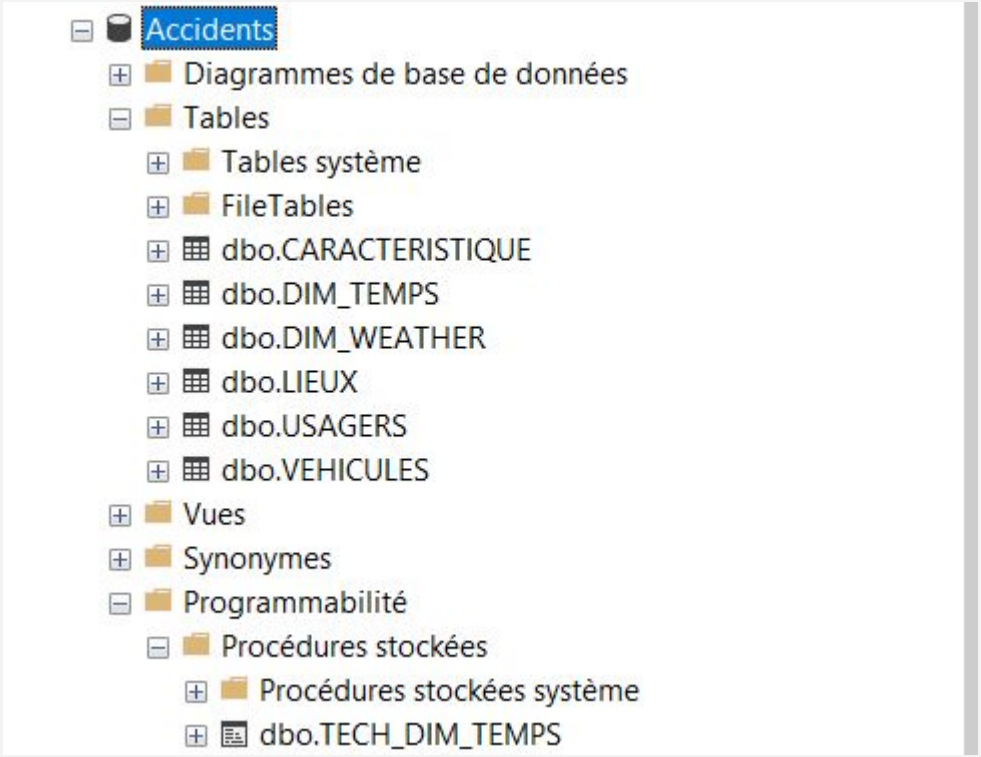
```java
private static String buildApiUrl(String date, String latitude, String longitude) {
    // Build the URL for the API request using the provided date, latitude, and longitude

    // Define the input and output date formats
    DateTimeFormatter inputFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    DateTimeFormatter outputFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

    // Parse the input date
    LocalDate dateParsed = LocalDate.parse(date, inputFormatter);

    // Format the date to the desired output format
    String firstDate = dateParsed.format(outputFormatter);

    // Increment the date by one day
    LocalDate incrementedDate = dateParsed.plusDays(1);
    String secondDate = incrementedDate.format(outputFormatter);

    return String.format("https://api.weatherbit.io/v2.0/history/hourly?lat=%s&lon=%s&start_date=%s&end_date=%s&key=%s", latitude, longitude, firstDate, secondDate, API_
}
```

ID_WEATHER	Date	Time	Temperature	Weather_Description	Wind_Speed	Wind_Direction	Atmospheric Pressure	Humidity	Dewpoint	Visibility
1	2008-03-08	15:00:00.0000000	29.0	Broken clouds	3.6	310	988	48	16.9	10
2	2008-03-08	16:00:00.0000000	29.0	Broken clouds	4.6	280	988	42	14.8	10
3	2008-03-08	17:00:00.0000000	30.0	Broken clouds	4.6	280	988	40	14.9	10
4	2008-03-08	18:00:00.0000000	29.0	Broken clouds	4.6	280	987	45	15.9	10
5	2008-03-08	19:00:00.0000000	28.0	Broken clouds	4.6	300	987	51	16.9	10
6	2008-03-08	20:00:00.0000000	24.0	Broken clouds	4.1	300	987	64	16.8	10
7	2008-03-08	21:00:00.0000000	23.0	Broken clouds	2.6	300	988	68	16.8	10
8	2008-03-08	22:00:00.0000000	21.0	Broken clouds	1.5	290	987	73	16.0	10
9	2008-03-08	23:00:00.0000000	22.0	Broken clouds	2.0	275	987	73	16.9	10
10	2008-03-08	00:00:00.0000000	23.0	Broken clouds	2.0	265	987	64	15.8	10

Extract transform load



Insertion des données dans la DB



Merci pour votre attention

Samar Neji
Mohamed Aziz Njaimi
Maroua Jbeli
Malik nairi
Hamza Ben Torkia
Khalil Khaled