Introduction

Dans un monde de plus en plus tourné vers la data-driven decision making, la gestion, l'intégration et l'analyse des données socio-économiques sont devenues des enjeux cruciaux pour comprendre les dynamiques mondiales. Que ce soit pour mesurer le développement humain, analyser les inégalités économiques ou étudier l'impact environnemental des pays, les données brutes issues de multiples sources nécessitent une structuration rigoureuse afin d'être exploitées de manière efficace.

Le projet présenté dans ce mémoire s'inscrit dans cette démarche : il a pour objectif de concevoir un **entrepôt de données (Data Warehouse)** permettant de centraliser, nettoyer et organiser des informations provenant de différentes bases de données publiques telles que l'indice de développement humain (HDI), les émissions de CO_2, le PIB par habitant et les sources d'énergie utilisées dans les pays du globe. À travers ce projet, nous mettons en œuvre une solution complète d'intégration et d'analyse décisionnelle, allant de l'extraction des données brutes jusqu'à la création de tableaux de bord interactifs via Power BI.

La démarche adoptée suit un processus ETL (Extract, Transform, Load) soigneusement pensé pour garantir la qualité, la cohérence et la fiabilité des données intégrées. Elle repose sur une modélisation en **schéma en étoile**, organisée autour d'une table de faits centrale soutenue par plusieurs tables de dimensions, facilitant ainsi l'interprétation et l'analyse multidimensionnelle.

Ce document se structure autour des grandes étapes du projet :

- Une présentation détaillée du **cahier des charges** et des objectifs d'analyse.
- Une explication de la **conception du Data Warehouse**, incluant la modélisation des dimensions et de la table de faits.
- La description du **processus ETL**, illustrant les phases d'extraction, de transformation et de chargement des données.
- Enfin, une analyse des résultats obtenus à travers des **visualisations BI** réalisées sous Power BI, permettant de répondre à des questions stratégiques liées au développement durable, aux inégalités économiques et aux politiques énergétiques mondiales.

À travers cette approche méthodique et structurée, ce projet illustre comment la Business Intelligence peut jouer un rôle clé dans la compréhension et la prise de décision sur des enjeux socio-économiques globaux.

Choix des données

Les données utilisées dans ce projet proviennent de la plateforme Our World in Data, qui agrège des indicateurs socio-économiques provenant de sources fiables telles que la **Banque mondiale**, le **FMI**, l'**ONU** et l'**Agence internationale de l'énergie**. Ces jeux de données ont été sélectionnés pour leur qualité, leur accessibilité et leur pertinence par rapport aux objectifs d'analyse.

Jeux de données utilisés :

- **Indice de développement humain (HDI)** : pour mesurer et classer le niveau de développement des pays.
- **Émissions de CO_2 par habitant** : pour analyser l'impact environnemental au fil du temps.
- **PIB par habitant** : pour évaluer la richesse économique des pays.
- **Sources d'énergie principales** : pour identifier les modes de production énergétique dominants.

- **Catégories de revenus** : pour segmenter les pays selon leur niveau de revenu.

Ces ensembles de données sont disponibles en format CSV, facilement exploitables dans un processus ETL. Leur mise à jour régulière et leur standardisation facilitent également la réalisation d'analyses pertinentes et comparatives.

Le choix de ces données permet de répondre à des questions stratégiques comme :

- Comment le développement humain varie-t-il selon les sources d'énergie ?
- Existe-t-il une corrélation entre le PIB et les émissions de CO_2 ?

En somme, ces jeux de données constituent une base solide pour la construction du **Data Warehouse** et la réalisation d'analyses décisionnelles via Power BI.

---

2. Architecture du projet

L'architecture du projet repose sur une approche simple, modulaire et adaptée au traitement de données structurées en volume limité. Elle est conçue pour garantir la cohérence, la scalabilité et la facilité d'exploitation des données tout au long du cycle ETL (Extraction, Transformation, Chargement) jusqu'à leur visualisation finale. L'architecture se décompose en trois grandes couches principales : **ETL**, **Stockage** et **Visualisation**.

---

a. Couche ETL (Extraction, Transformation, Chargement)

La première couche de l'architecture concerne le processus ETL, qui permet de préparer les données brutes pour leur intégration dans le Data Warehouse. Les étapes clés sont les suivantes :

1. **Extraction** :
    - Les données brutes sont extraites depuis des fichiers CSV stockés dans le répertoire `data/input/`. Ces fichiers proviennent de la plateforme **Our World in Data** et contiennent des informations socio-économiques telles que l'indice de développement humain (HDI), les émissions de CO_2 par habitant, le PIB par habitant, les sources d'énergie principales et les catégories de revenus.
    - Le script d'extraction (`extractor.py`) récupère ces fichiers et les charge en mémoire sous forme de structures de données Python (par exemple, pandas DataFrame).
2. **Transformation** :
    - Une étape de transformation est réalisée pour nettoyer, normaliser et enrichir les données. Cette étape inclut :
        - **Nettoyage** : Suppression des valeurs manquantes, correction des erreurs typographiques ou formatives.
        - **Normalisation** : Conversion des unités de mesure (par exemple, standardisation des noms des pays).
        - **Enrichissement** : Création de nouvelles colonnes ou indicateurs, tels que la catégorie de revenu basée sur le PIB par habitant ou la classification de l'HDI.

        - **Catégorisation** : Attribution de niveaux de développement humain (très élevé, élevé, moyen, faible) et de catégories de revenus (riche, moyen supérieur, moyen inférieur, pauvre).
    - La classe `DataTransformer` dans `transformer.py` gère cette étape, appliquant des règles spécifiques définies dans le fichier de configuration (`config.py`).

3. **Chargement** :

    - Les données transformées sont sauvegardées dans des fichiers CSV dans le répertoire `data/output/`. Ces fichiers servent ensuite à charger les tables du Data Warehouse.
    - Le script de chargement (`loader.py`) génère les fichiers finaux nécessaires pour chaque table du modèle étoile.

---

b. Couche Stockage

La deuxième couche de l'architecture concerne le stockage des données transformées dans un environnement relationnel. Pour ce projet, une base de données **PostgreSQL** a été choisie pour son efficacité, sa robustesse et sa compatibilité avec les outils d'analyse comme Power BI.

1. **Base de données PostgreSQL** :

    - Une instance PostgreSQL a été créée pour héberger les tables du Data Warehouse. Le schéma utilisé suit un modèle étoile, composé d'une table de faits centrale (`Fait_SocioEconomique`) entourée de plusieurs tables de dimensions (`Dimension_Pays`, `Dimension_Temps`, `Dimension_Energy`, `Dimension_Developpement_Humain`, `Dimension_Revenus`).
    - Les relations entre les tables sont définies via des clés primaires et étrangères, garantissant l'intégrité des données.

2. **Hébergement avec Docker** :

    - Pour assurer une mise en place rapide, isolée et facilement réplicable, PostgreSQL est hébergé dans un conteneur Docker. Cela permet de configurer l'environnement de base de données de manière uniforme, indépendamment de l'infrastructure hôte.
    - Un fichier `docker-compose.yml` peut être utilisé pour démarrer l'instance PostgreSQL avec des configurations prédéfinies (port, nom de base de données, etc.).

---

c. Couche Visualisation

La troisième couche de l'architecture concerne la visualisation des données via **Power BI**. Ce logiciel permet de créer des rapports dynamiques et interactifs, facilitant l'exploration et l'interprétation des résultats.

1. **Connexion à la base de données** :

    - Power BI se connecte directement à la base de données PostgreSQL via un pilote ODBC ou un connecteur natif.
    - Les tables du Data Warehouse sont importées dans Power BI, où elles sont modélisées pour refléter les relations existantes dans le schéma étoile.

2. **Modélisation des relations** :
    - Les relations entre les tables sont définies dans Power BI pour garantir une exploration cohérente des données. Par exemple, la table de faits (`Fait_SocioEconomique`) est liée aux tables de dimensions via leurs clés primaires et étrangères.
3. **Création de rapports visuels** :
    - Des visualisations interactives sont créées pour explorer différents aspects des données, tels que :
        - Évolution des émissions de CO_2 par pays au fil du temps.
        - Corrélation entre le PIB par habitant et le niveau de développement humain.
        - Répartition des sources d'énergie principales selon les régions géographiques.
    - Les indicateurs clés (KPIs) sont mis en avant grâce à des graphiques, des cartes choroplèthes et des tableaux dynamiques.

Objectif global de l'architecture

L'objectif principal de cette architecture est de garantir une gestion cohérente et évolutive du cycle de vie des données, de leur ingestion jusqu'à leur exploitation analytique. En séparant clairement les responsabilités entre les différentes couches (ETL, Stockage, Visualisation), le projet assure :

- **Flexibilité** : Facilité d'ajout ou de modification de nouvelles sources de données.
- **Scalabilité** : Capacité à évoluer en fonction de l'augmentation du volume des données.
- **Reproductibilité** : Mise en œuvre simplifiée grâce à l'utilisation de Docker pour l'environnement de base de données.
- **Interprétabilité** : Présentation des résultats sous forme de visualisations intuitives et interactives via Power BI.

Le fait Socio-Économique

Dans le cadre de ce projet, la table de faits principale est `Fait_SocioEconomique`, qui centralise les mesures socio-économiques clés liées aux pays et aux années. Cette table joue un rôle crucial dans l'analyse des dynamiques globales en combinant des indicateurs tels que l'indice de développement humain (HDI), les émissions de CO_2 par habitant, le PIB par habitant et les sources d'énergie principales.

Rôle du Fait Socio-Économique

La table `Fait_SocioEconomique` sert de cœur du modèle étoile, connectant les différentes dimensions pour permettre une analyse multidimensionnelle des données. Elle contient les mesures numériques (faits) qui sont analysées en relation avec les contextes fournis par les tables de dimensions.

Structure de la Table `Fait_SocioEconomique`

Voici une description détaillée des champs présents dans la table `Fait_SocioEconomique` :

Champ	Type de Donnée	Description

Champ	Type de Donnée	Description
`ID_Pays`	integer	Clé étrangère vers la table `Dimension_Pays`.
`ID_Annee`	integer	Clé étrangère vers la table `Dimension_Temps`.
`ID_Revenu`	integer	Clé étrangère vers la table `Dimension_Revenus`.
`ID_energie_principale`	string	Clé étrangère vers la table `Dimension_Energy`.
`ID_niveau_developpement`	string	Clé étrangère vers la table `Dimension_Developpement_Humain`.
`HDI`	float	Indice de développement humain pour le pays et l'année spécifiés.
`Emissions_CO2_par_habitant`	float	Émissions de dioxyde de carbone par habitant pour le pays et l'année spécifiés.
`PIB_par_habitant`	float	Produit intérieur brut par habitant pour le pays et l'année spécifiés.

Dimensions Associées à la Table `Fait_SocioEconomique`

Les dimensions fournissent des informations contextuelles sur les faits. Voici une description détaillée des dimensions utilisées :

Dimension	Contenu	Utilisation dans l'analyse
Dimension_Pays	Informations sur les pays : nom, code ISO, région géographique.	Analyser les performances socio-économiques par pays ou par région.
Dimension_Temps	Année, mois, jour.	Réaliser des analyses temporelles précises (tendances au fil du temps).
Dimension_Energy	Sources d'énergie principales (pétrole, gaz, électricité, etc.).	Identifier les stratégies énergétiques dominantes et leurs implications environnementales.
Dimension_Developpement_Humain	Niveaux de développement humain (très élevé, élevé, moyen, faible).	Comparer les niveaux de développement entre les pays et analyser leur évolution.

Dimension	Contenu	Utilisation dans l'analyse
Dimension_Revenus	Catégories de revenus basées sur le PIB par habitant (riche, moyen supérieur, moyen inférieur, pauvre).	Segmenter les pays selon leur richesse économique et observer les inégalités.

Exemple d'utilisation de la Table `Fait_SocioEconomique`

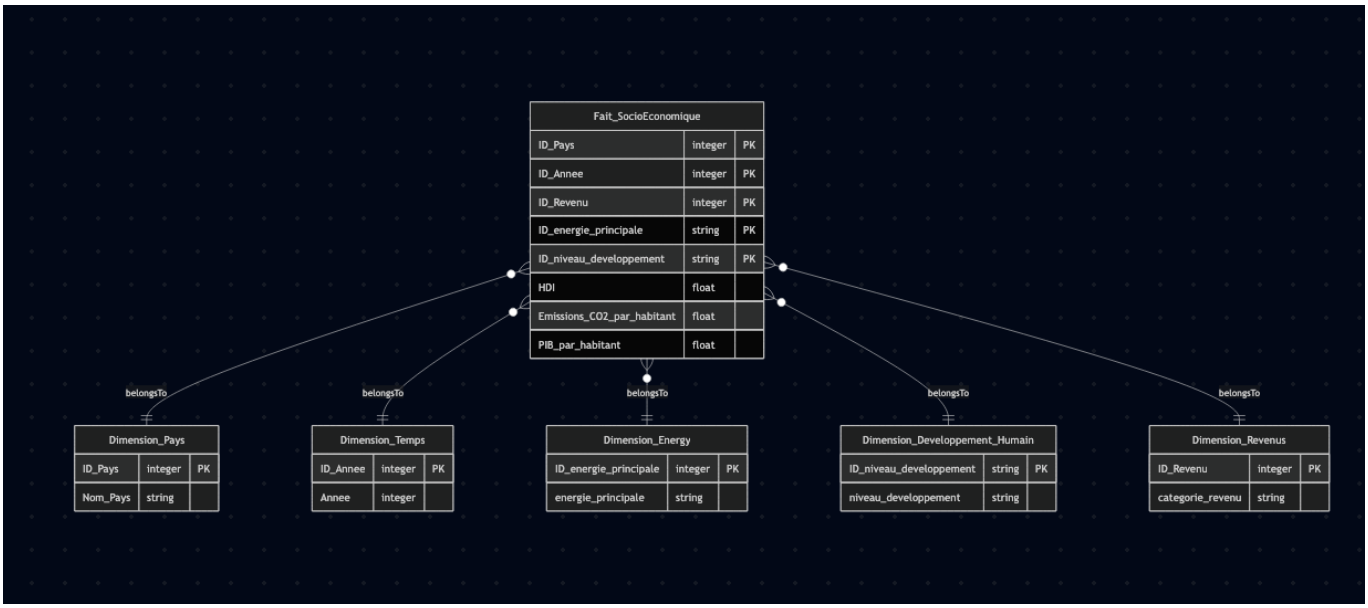
En combinant les dimensions et les faits, il est possible d'effectuer des analyses approfondies. Par exemple :

1. **Analyse de la corrélation entre le PIB par habitant et les émissions de CO_2** :
    - Utiliser les dimensions `Dimension_Pays` et `Dimension_Temps` pour filtrer les données par pays et par période.
    - Observer la tendance des émissions de CO_2 (`Emissions_CO2_par_habitant`) en fonction du PIB par habitant (`PIB_par_habitant`).
2. **Comparaison des niveaux de développement humain par région géographique** :
    - Filtrer les données par région géographique via la dimension `Dimension_Pays`.
    - Analyser les valeurs d'HDI (`HDI`) pour chaque pays et comparer les résultats entre les régions.
3. **Identification des sources d'énergie dominantes par niveau de revenu** :
    - Utiliser la dimension `Dimension_Revenus` pour segmenter les pays selon leur catégorie de revenu.
    - Observer les sources d'énergie principales (`ID_energie_principale`) pour chaque catégorie de revenu.

Conclusion

La table `Fait_SocioEconomique` constitue le cœur du modèle étoile, offrant une structure robuste pour intégrer et analyser des données socio-économiques complexes. En combinant les dimensions pertinentes, elle permet d'explorer des relations multidimensionnelles entre les indicateurs clés, telles que le développement humain, les émissions de CO_2, le PIB et les stratégies énergétiques. Cela facilite la réalisation d'analyses approfondies et l'élaboration de visualisations décisionnelles riches via Power BI.

Modèle en étoile



1. Approche et Méthodologie

Le projet repose sur une approche structurée d'**Extraction, Transformation et Chargement (ETL)** des données socio-économiques provenant de fichiers CSV bruts. L'objectif principal est de produire des jeux de données propres, structurés et conformes à un modèle étoile, prêts à être intégrés dans un **Data Warehouse relationnel** pour une analyse décisionnelle avancée via Power BI.

Architecture du projet

Le projet suit une architecture modulaire claire, facilitant la maintenance, l'évolutivité et la compréhension globale du processus ETL. Voici sa structure principale :

```
BI-python/
├── data/
│   ├── input/              # Fichiers CSV bruts
│   └── output/             # Fichiers CSV transformés
├── src/
│   ├── __init__.py         # Marque le répertoire 'src' comme package
Python
│   ├── config.py           # Configuration des chemins et seuils
│   ├── extractor.py        # Extraction des données
│   ├── transformer.py      # Transformation des données
│   └── loader.py           # Chargement des CSV finaux ET chargement
dans PostgreSQL
├── requirements.txt        # Liste des dépendances Python
└── run_etl.py              # Point d'entrée du script
```

Cette organisation permet une séparation claire entre les différentes phases du pipeline ETL : extraction, transformation et **chargement final dans une base de données SQL**.

Technologies utilisées

Le projet est développé en **Python 3.8+**, grâce à ses capacités puissantes de traitement de données via la bibliothèque **pandas**. Les choix technologiques ont été guidés par la simplicité, la rapidité de développement et la portabilité du code.

Les principales bibliothèques utilisées sont :

- **pandas** : manipulation et transformation des données.
- **os / glob** : gestion des chemins et fichiers.
- **logging** : journalisation des étapes du processus ETL.
- **psycopg2** : connexion et insertion des données dans PostgreSQL.

Phases du pipeline ETL

1. Extraction

La première phase consiste à charger les données brutes depuis les fichiers CSV situés dans le dossier `data/input/`. Ces fichiers proviennent de sources fiables telles que Our World in Data, qui agrège des données issues d'institutions internationales (Banque mondiale, ONU, etc.).

Les quatre jeux de données principaux extraits sont :

- `human-development-index.csv` : Indice de développement humain (HDI).
- `electricity-prod-source-stacked.csv` : Répartition des sources d'énergie.
- `gdp-per-capita-worldbank.csv` : PIB par habitant.
- `co-emissions-per-capita.csv` : Émissions de CO_2 par habitant.

L'extraction est gérée par la classe `DataExtractor` définie dans le fichier `extractor.py`.

2. Transformation

Les données extraites sont nettoyées, transformées et enrichies afin de respecter les exigences du modèle étoile. Cette étape inclut notamment :

- La catégorisation de l'indice de développement humain (`categorize_hdi`) selon des seuils configurables.
- L'identification de l'énergie principale (`determine_primary_energy`) par pays et par année.
- La catégorisation des revenus (`categorize_income`) basée sur le PIB par habitant.
- Le filtrage des données pour ne conserver que les années supérieures ou égales à 1990.
- La création des tables dimensionnelles (`Dimension_Pays`, `Dimension_Temps`, etc.) à partir des données consolidées.
- La génération de la table de faits finale (`Fait_SocioEconomique`).

Toutes ces opérations sont centralisées dans la classe `DataTransformer` implémentée dans `transformer.py`.

3. Chargement

Le fichier `loader.py` remplit **deux rôles essentiels** :

1. **Génération des fichiers CSV finaux** : les tables transformées (dimensions et faits) sont sauvegardées sous forme de fichiers CSV dans le répertoire `data/output/`.
    - `Dimension_Pays.csv`
    - `Dimension_Temps.csv`
    - `Dimension_Energy.csv`
    - `Dimension_Developpement_Humain.csv`
    - `Dimension_Revenus.csv`
    - `Fait_SocioEconomique.csv`
2. **Chargement des données dans PostgreSQL** : une fois les fichiers générés, ils sont automatiquement insérés dans une base de données PostgreSQL. Cette étape garantit que le processus d'intégration des données soit complet de bout en bout.

Le chargement dans la base SQL peut se faire via des requêtes SQL pures ou avec l'aide de bibliothèques comme **pandas.DataFrame.to_sql()** ou **psycopg2**.

Configuration du projet

Les paramètres clés du projet (seuils HDI, seuils de revenus, chemins vers les fichiers) sont externalisés dans le fichier `config.py`. Cela permet une configuration flexible sans avoir à modifier le cœur du code.

Exemple de configuration :

```
# Seuils pour la catégorisation de l'HDI
HDI_THRESHOLDS = {
    "Très élevé": 0.800,
    "Élevé": 0.700,
    "Moyen": 0.550,
    "Faible": 0.000,
}

# Catégories de revenus basées sur le PIB par habitant
INCOME_CATEGORIES = {
    "riche": (20000, float("inf")),
    "moyen_sup": (10000, 20000),
    "moyen_inférieur": (5000, 10000),
    "pauvres": (0, 5000),
}
```

Avantages de l'approche adoptée

- **Intégration complète de bout en bout** : du fichier CSV brut jusqu'à l'intégration dans PostgreSQL.
- **Modularité** : Chaque étape du pipeline est encapsulée dans une classe spécifique, facilitant la réutilisation et le test unitaire.
- **Traçabilité** : Des logs sont générés à chaque étape, permettant de suivre précisément l'exécution du processus ETL.

- **Simplicité** : Aucune dépendance complexe n'est requise, rendant le projet facile à exécuter et à reproduire.
- **Extensibilité** : Le projet peut facilement être enrichi avec de nouvelles sources de données ou des transformations supplémentaires.

2. Objectif du Nettoyage des Données

Avant toute intégration dans la base de données PostgreSQL, il était indispensable d'assurer une qualité optimale des données en corrigeant les problèmes fréquents liés aux jeux de données bruts. Le nettoyage et la préparation des données ont été réalisés pour garantir leur cohérence, leur fiabilité et leur conformité avec le modèle étoile utilisé pour l'analyse socio-économique.

Problèmes courants identifiés

Les jeux de données bruts provenant de sources externes (Our World in Data) pouvaient présenter plusieurs défis, tels que :

- **Valeurs manquantes** : Certaines lignes ou colonnes pouvaient contenir des valeurs `NaN` ou vides, ce qui aurait pu perturber les analyses ultérieures.
- **Formats non standardisés** : Les noms des pays, les dates ou les unités de mesure pouvaient varier entre les fichiers, nécessitant une normalisation.
- **Doublons** : Des entrées redondantes ou mal orthographiées pouvaient exister, entraînant des erreurs lors de l'agrégation des données.
- **Incohérences** : Certains pays ou années pouvaient être inconsistants entre les différentes sources de données (par exemple, des émissions de CO_2 manquantes pour certaines années).
- **Données non exploitables** : Certains indicateurs comme le PIB par habitant ou l'HDI pouvaient ne pas être disponibles pour toutes les entités ou périodes.

Objectifs spécifiques du nettoyage

Pour répondre à ces défis, les objectifs du nettoyage des données étaient les suivants :

1. **Suppression des valeurs manquantes critiques** :
    - Identifier et supprimer les lignes où des valeurs essentielles (comme le nom du pays, l'année, le PIB par habitant ou l'HDI) sont manquantes.
    - Pour les champs moins critiques, des stratégies d'imputation ou de suppression sélective ont été appliquées selon leur importance.
2. **Uniformisation des formats** :
    - Mise en minuscule des noms des pays pour éviter les doublons liés à la casse.
    - Normalisation des dates sous un format standardisé (`YYYY-MM-DD`).
    - Conversion des nombres en types numériques appropriés (floats ou integers).
3. **Création d'identifiants uniques** :
    - Génération d'identifiants uniques pour les principales entités (pays, années, sources d'énergie, etc.) afin de garantir l'intégrité des relations dans la base de données.

- Par exemple, chaque pays reçoit un `ID_Pays`, chaque année un `ID_Annee`, etc., facilitant les jointures dans le modèle étoile.

4. **Détection et traitement des incohérences** :
   - Vérification des correspondances entre les tables (par exemple, les pays mentionnés dans différentes sources doivent être cohérents).
   - Suppression des doublons ou résolution des conflits lors de la fusion de données provenant de plusieurs sources.
   - Validation que les émissions de CO_2 par habitant correspondent bien aux années spécifiées.
   - Vérification que les catégories de revenus basées sur le PIB respectent les seuils définis dans la configuration (`config.py`).
5. **Création de nouvelles tables à partir de la table principale** :
   - Extraction des informations contextuelles pour créer des tables dimensionnelles distinctes. Par exemple :
     - La table `Dimension_Temps` a été créée à partir des colonnes relatives aux années.
     - La table `Dimension_Energy` a été générée pour capturer les sources d'énergie principales utilisées par chaque pays.
6. **Normalisation des textes** :
   - Suppression des espaces superflus.
   - Correction des erreurs d'encodage (caractères spéciaux mal interprétés).
   - Standardisation des noms des pays et des régions géographiques.
7. **Validation des règles métier** :
   - Vérification que les émissions de CO_2 par habitant correspondent bien aux années spécifiées.
   - Validation que les catégories de revenus basées sur le PIB respectent les seuils définis dans la configuration (`config.py`).

Exemple concret de nettoyage

Voici quelques exemples concrets des opérations de nettoyage effectuées :

Type de nettoyage	Action réalisée
Suppression des valeurs manquantes	Lignes contenant des valeurs manquantes critiques (Pays, Année, HDI, etc.) ont été exclues.
Uniformisation des formats	Les noms des pays ont été mis en minuscule pour éviter les doublons.
Création d'identifiants uniques	Chaque pays a reçu un `ID_Pays` unique, chaque année un `ID_Annee`, etc.
Détection des doublons	Les doublons dans les noms des pays ou des années ont été corrigés ou supprimés.

Type de nettoyage	Action réalisée
Catégorisation automatique	Les niveaux de développement humain (HDI) et les catégories de revenus ont été calculés automatiquement.

Résultats du nettoyage

Après le processus de nettoyage et de transformation, les données ont été structurées de manière à respecter les exigences du modèle étoile :

- Les dimensions (`Dimension_Pays`, `Dimension_Temps`, etc.) contiennent des informations contextuelles claires et uniformes.
- La table de faits (`Fait_SocioEconomique`) est enrichie avec des clés étrangères valides, garantissant des relations cohérentes avec les dimensions.
- Les mesures (HDI, émissions de CO_2, PIB par habitant, etc.) sont prêtes à être analysées via Power BI.

3. Détails des Scripts d'Intégration

Great! Based on the scripts you've provided (`extractor.py`, `transformer.py`, and `loader.py`), I can help you analyze and explain their functionalities in detail. Below is a comprehensive breakdown of each script, highlighting their responsibilities and how they fit into the overall ETL pipeline.

---

1. `extractor.py`: Extraction des Données Brutes

Fonctionnalités principales:

Le fichier `extractor.py` est responsable de l'extraction des données brutes à partir des fichiers CSV situés dans le répertoire `data/input/`. Il encapsule la logique d'importation des données et les rend disponibles pour les étapes suivantes du processus ETL.

Classe `DataExtractor`:

- **Attributs** :
  - `input_dir`: Chemin vers le répertoire contenant les fichiers CSV bruts.
- **Méthodes** :
  1. `read_csv(filename)` :
     - Lit un fichier CSV spécifique et retourne un DataFrame pandas.
     - Utilise `os.path.join` pour construire le chemin complet du fichier.
     - Exemple : `self.read_csv("human-development-index.csv")`.
  2. `extract_all()` :
     - Charge tous les fichiers CSV nécessaires et les stocke dans un dictionnaire.
     - Les clés du dictionnaire correspondent aux noms des jeux de données (par exemple, `"hdi"`, `"energy"`, `"gdp"`, `"co2"`).
     - Exemple de sortie :

```
{
    "hdi": df_hdi,
```

```
        "energy": df_energy,
        "gdp": df_gdp,
        "co2": df_co2
    }
```

Exemple d'utilisation :

```
from src.extractor import DataExtractor

# Initialisation
extractor = DataExtractor()

# Extraction de tous les fichiers CSV
data = extractor.extract_all()
```

---

2. `transformer.py`: Transformation des Données

Fonctionnalités principales:

Le fichier `transformer.py` contient les fonctions nécessaires pour nettoyer, transformer et enrichir les données extraites. Cette étape garantit que les données soient conformes au modèle étoile final.

Classe `DataTransformer`:

- **Attributs** :
  - Aucun attribut spécifique n'est défini dans cette classe.
- **Méthodes** :
  1. `categorize_hdi(df_hdi)` :
     - Catégorise l'Indice de Développement Humain (HDI) en fonction des seuils définis dans `config.py`.
     - Utilise `pd.cut` pour diviser les valeurs d'HDI en catégories ("Faible", "Moyen", "Élevé", "Très élevé").
     - Ajoute une nouvelle colonne `niveau_developpement` au DataFrame.
  2. `determine_primary_energy(df_energy)` :
     - Détermine l'énergie principale utilisée par chaque pays/année en calculant la part relative de chaque source d'énergie.
     - Crée une nouvelle colonne `energie_principale` avec les valeurs possibles : `"fossil"`, `"nuclear"`, `"renouvelable"`, ou `"mix"`.
  3. `categorize_income(df_gdp)` :
     - Catégorise les revenus basés sur le PIB par habitant en fonction des seuils définis dans `config.py`.
     - Utilise `pd.cut` pour diviser les valeurs de PIB par habitant en catégories ("riche", "moyen_sup", "moyen_inférieur", "pauvres").
     - Ajoute une nouvelle colonne `categorie_revenu` au DataFrame.
  4. `filter_co2(df_co2)` :

- Filtre les émissions de CO_2 pour ne conserver que les données postérieures à 1990.
- Supprime les lignes où les valeurs d'émissions de CO_2 sont manquantes.

5. `create_dimensions(df_hdi, df_energy, df_gdp, df_co2)` :
   - Crée les tables dimensionnelles (`Dimension_Pays`, `Dimension_Temps`, etc.) et la table de faits (`Fait_SocioEconomique`) à partir des données transformées.
   - Génère des identifiants uniques pour chaque entité (pays, année, énergie, etc.).

Exemple d'utilisation :

```
from src.transformer import DataTransformer

# Initialisation
transformer = DataTransformer()

# Transformation des données
df_hdi_transformed = transformer.categorize_hdi(data["hdi"])
df_energy_transformed =
transformer.determine_primary_energy(data["energy"])
df_gdp_transformed = transformer.categorize_income(data["gdp"])
df_co2_transformed = transformer.filter_co2(data["co2"])

# Création des dimensions et de la table de faits
dimensions = transformer.create_dimensions(
    df_hdi_transformed,
    df_energy_transformed,
    df_gdp_transformed,
    df_co2_transformed
)
```

---

3. `loader.py`: Chargement des Données

Fonctionnalités principales:

Le fichier `loader.py` gère deux responsabilités principales :

1. Sauvegarde des données transformées sous forme de fichiers CSV dans le répertoire `data/output/`.
2. Chargement des données dans une base de données PostgreSQL.

Classe `DataLoader`:

- **Attributs** :
  - `output_dir`: Chemin vers le répertoire de sortie pour les fichiers CSV.
- **Méthodes** :
  1. `save_to_csv(df, filename)` :
     - Sauvegarde un DataFrame dans un fichier CSV.
     - Utilise `df.to_csv` avec l'encodage UTF-8.

2. `load_all(dimensions, df_faits)` :
    - Sauvegarde toutes les tables dimensionnelles et la table de faits dans des fichiers CSV respectifs.
    - Nomme les fichiers selon leur rôle (ex. `Dimension_Pays.csv`, `Fait_SocioEconomique.csv`).

Classe `DatabaseLoader`:

- **Attributs** :
    - `db_config`: Configuration de la connexion à la base de données PostgreSQL (chargée depuis `.env`).
- **Méthodes** :
    1. `connect()` :
        - Établit une connexion à la base de données PostgreSQL.
    2. `create_tables()` :
        - Crée les tables nécessaires dans la base de données si elles n'existent pas déjà.
        - Utilise des requêtes SQL `CREATE TABLE IF NOT EXISTS`.
    3. `truncate_table(table_name)` :
        - Vide une table existante avant de charger de nouvelles données.
    4. `load_csv_to_table(file_path, table_name)` :
        - Charge un fichier CSV dans une table PostgreSQL.
        - Vérifie l'encodage du fichier et s'assure que les colonnes attendues sont présentes.
        - Utilise `cursor.copy_from` pour charger efficacement les données.
    5. `close()` :
        - Ferme la connexion à la base de données.

Visualisation avec Power BI

Avant de pouvoir établir la connexion entre Power BI et la base de données PostgreSQL, il a été nécessaire de lancer le conteneur Docker. Ce conteneur héberge la base de données créée spécifiquement pour le projet et doit être en cours d'exécution pour que Power BI puisse y accéder. Une fois le conteneur démarré, la connexion aux données peut être réalisée sans difficulté.

Cette étape est essentielle car elle garantit que l'environnement PostgreSQL est opérationnel et accessible via le réseau local.

---

1. Connexion à la Base de Données PostgreSQL

La connexion entre Power BI et PostgreSQL a été effectuée en utilisant le connecteur natif PostgreSQL disponible dans Power BI Desktop. Les étapes suivies ont été les suivantes :

1. **Lancement du conteneur Docker pour activer PostgreSQL** :
    - Le conteneur Docker héberge une instance PostgreSQL configurée spécifiquement pour ce projet.
    - L'adresse IP et le port du conteneur sont définis dans le fichier `docker-compose.yml` ou directement lors du lancement du conteneur.
    - Exemple de commande pour lancer le conteneur :

```
docker-compose up -d
```

2. **Utilisation des informations de connexion correctes** :
   - Adresse IP du conteneur (généralement `localhost` ou `127.0.0.1`).
   - Port PostgreSQL (par défaut : `5432`).
   - Nom de la base de données (`etl_project`).
   - Identifiants d'accès (utilisateur et mot de passe).
3. **Sélection et importation des tables principales du projet** :
   - Dans Power BI Desktop, un nouveau rapport est créé.
   - Le connecteur PostgreSQL est sélectionné, et les détails de connexion sont renseignés.
   - Les tables dimensionnelles et la table de faits sont sélectionnées :
     - `Dimension_Pays`
     - `Dimension_Temps`
     - `Dimension_Energy`
     - `Dimension_Developpement_Humain`
     - `Dimension_Revenus`
     - `Fait_SocioEconomique`
4. **Reconstitution de la structure du modèle relationnel** :
   - Après l'importation des tables, Power BI permet de reconstituer les relations prévues dans le modèle étoile.
   - Les clés primaires et étrangères sont définies manuellement pour garantir l'intégrité des relations entre les tables.
   - Par exemple :
     - La table `Fait_SocioEconomique` est liée à `Dimension_Pays` via la colonne `ID_Pays`.
     - La table `Fait_SocioEconomique` est liée à `Dimension_Temps` via la colonne `ID_Annee`.

---

2. Création des Visualisations

Une fois les données chargées et les relations établies, Power BI permet de créer des visualisations interactives pour explorer les données socio-économiques. Voici quelques exemples d'analyses possibles :

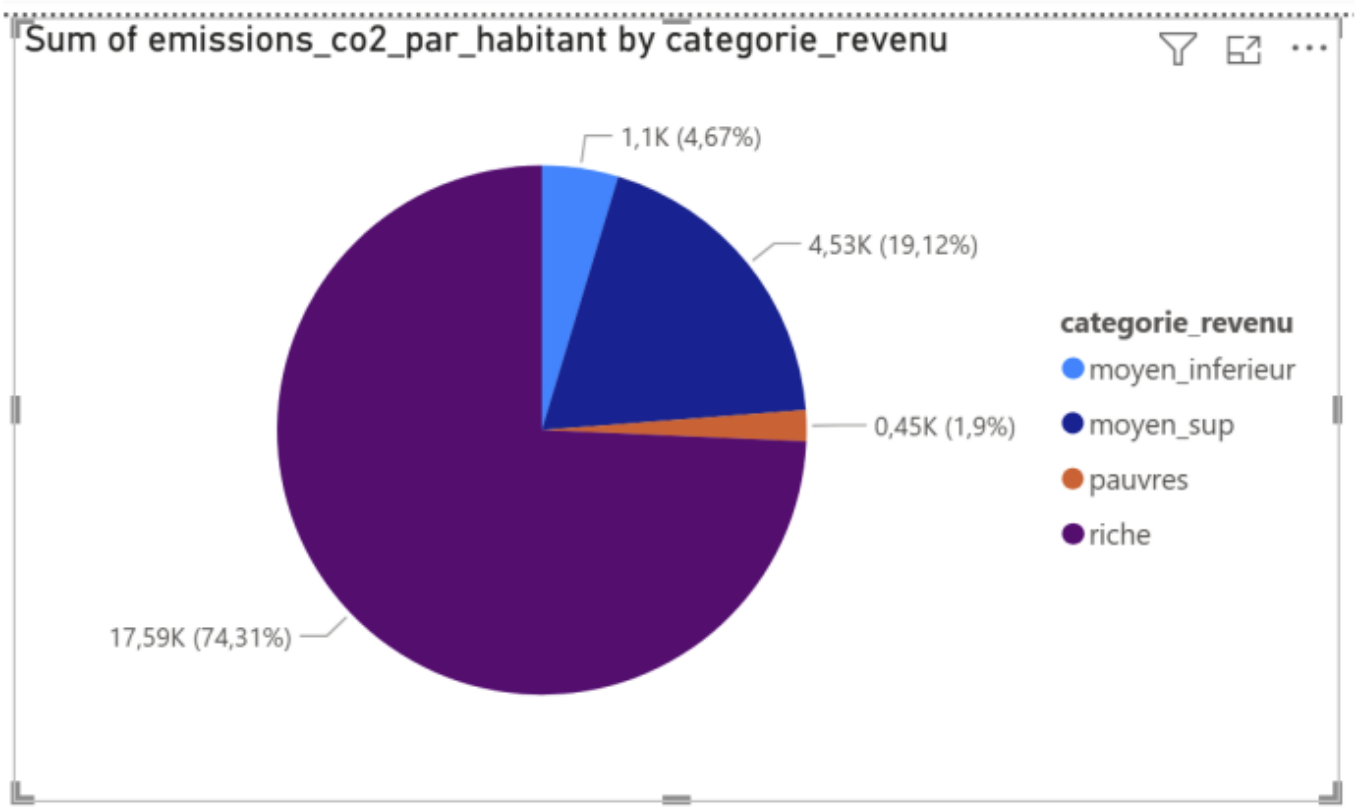
1. **Analyse des tendances temporelles** :
   - Utilisation de graphiques linéaires ou cartes choroplèthes pour visualiser l'évolution des indicateurs au fil du temps.
   - Exemple : Évolution des émissions de CO_2 par habitant par pays sur plusieurs années.
2. **Comparaison des pays selon différents indicateurs** :
   - Cartes géographiques pour comparer les niveaux de développement humain, les sources d'énergie principales et les catégories de revenus.

- Tableaux dynamiques pour afficher les valeurs moyennes de PIB par habitant et d'HDI par région géographique.

3. **Analyse multidimensionnelle** :
   - Dashboards interactifs permettant de filtrer les données selon différentes dimensions (pays, année, catégorie de revenu, etc.).
   - Utilisation de KPIs pour mettre en avant des indicateurs clés tels que le HDI moyen mondial ou les émissions de CO_2 totales.
4. **Identification des corrélations** :
   - Graphiques à bulles ou matrices de corrélation pour analyser les relations entre les indicateurs socio-économiques.
   - Exemple : Corrélation entre le PIB par habitant et les émissions de CO_2 par habitant.

Observations et analyses

Répartition des Émissions par Catégorie de Revenu



Observations

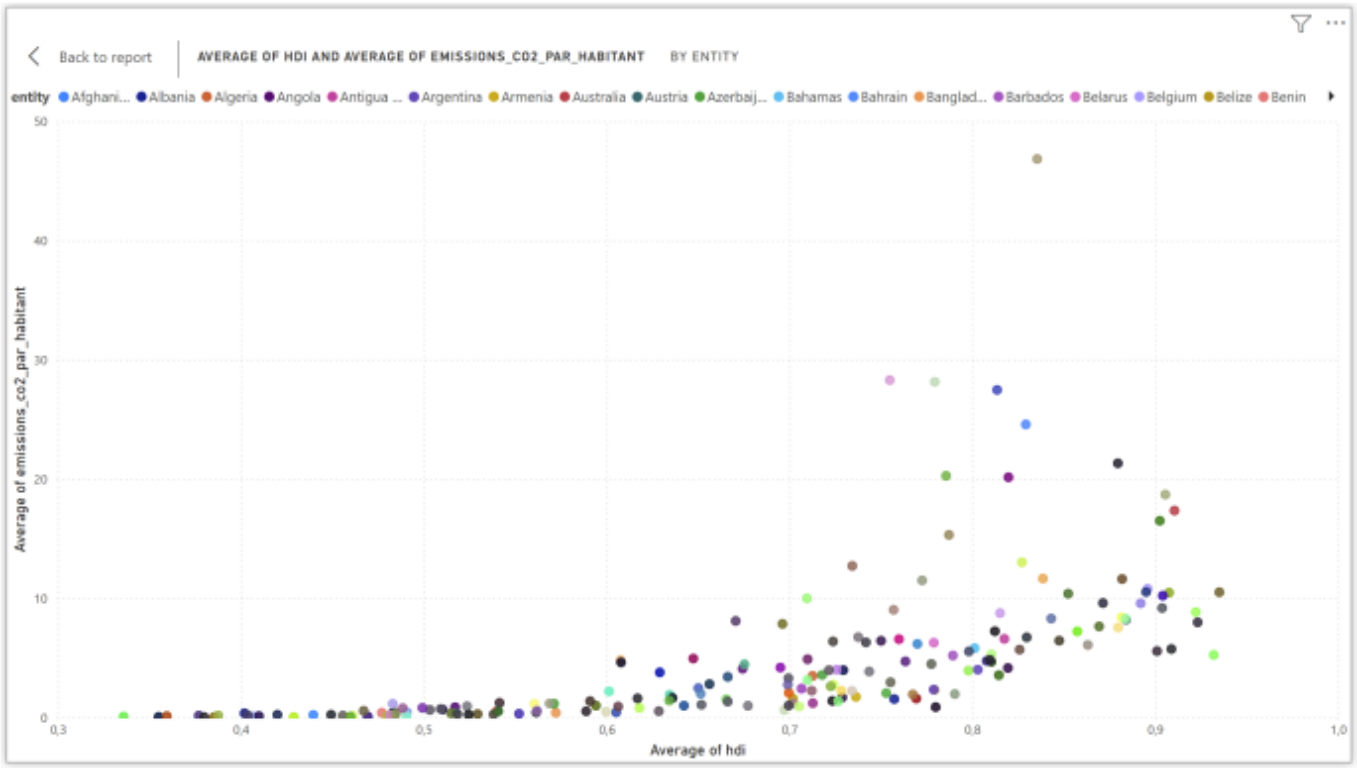
- **Pays Riches** : Les pays riches ("Riche") dominent les émissions (74.31 %). Cela s'explique par leur consommation élevée d'énergie, leur industrialisation avancée, et leur mode de vie intensif en ressources.
- **Pays Moyen-Revenu** : Les pays à revenu "Moyen%sup" (19.12 %) et "Moyen%inferieur" (4.67 %) contribuent également aux émissions, mais dans une moindre mesure. Ces pays se trouvent souvent dans des phases de développement industriel où l'utilisation d'énergies fossiles augmente.

- **Pays Pauvres** : Les pays pauvres ("Pauvres") ont une contribution minime (1.9 %). Leur faible accès à l'énergie et leur économie moins industrialisée expliquent cette faible empreinte carbone.

Interprétation

- **Modèle de Consommation** : Les pays riches ont un modèle de consommation énergétique très carboné, tandis que les pays pauvres n'ont pas encore atteint un niveau de consommation suffisant pour produire des émissions significatives.
- **Transition Énergétique** : Il est crucial de soutenir les pays à revenu moyen dans leur transition vers des énergies propres avant qu'ils ne suivent le même chemin que les pays riches.

Corrélation entre l'indice de développement humain et Émissions



1. Observations

Le scatter plot montre la relation entre l'**Indice de Développement Humain (HDI)** et les **émissions de CO_2 par habitant** pour différents pays. Voici les principales observations :

- **Répartition générale** :
  - Les points sont dispersés sur le graphique, indiquant une variabilité dans les relations entre HDI et émissions de CO_2.
  - Certains pays présentent des émissions élevées malgré un HDI relativement faible, tandis que d'autres ont des émissions faibles avec un HDI élevé.
- **Clusters distincts** :
  - À gauche du graphique (HDI bas), les émissions de CO_2 sont généralement faibles, ce qui est logique car ces pays sont souvent moins industrialisés.

    - À droite du graphique (HDI élevé), on observe une dispersion plus large des émissions de CO_2. Certains pays très développés (HDI élevé) émettent beaucoup de CO_2, tandis que d'autres limitent leurs émissions.

- **Corrélation globale** :
    - Une tendance légèrement positive semble exister entre le HDI et les émissions de CO_2. Cependant, cette corrélation n'est pas linéaire ou stricte, car certains pays à HDI élevé affichent des émissions relativement basses.

- **Exemples notables** :
    - Certains pays situés en haut à droite du graphique (HDI élevé et émissions élevées) peuvent être identifiés comme des économies industrielles ou pétrolières.
    - Des pays en bas à gauche (HDI bas et émissions basses) représentent souvent des économies en développement ou moins industrialisées.

Interprétations

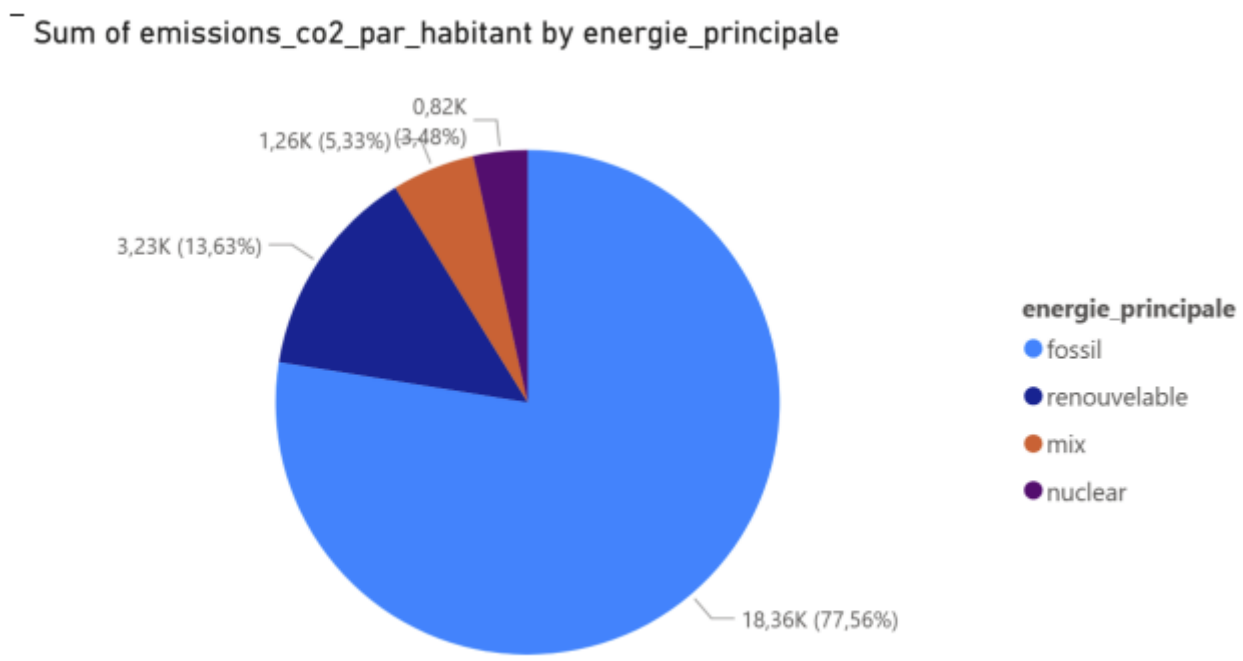
- **Développement économique et émissions de CO_2** :
    - Les pays à HDI élevé tendent à avoir des économies plus industrialisées et consomment davantage d'énergie, ce qui peut entraîner des émissions de CO_2 plus importantes. Cependant, certains pays riches investissent également dans des technologies vertes et des politiques environnementales strictes, réduisant leur empreinte carbone.

- **Effets des politiques environnementales** :
    - La dispersion des points à HDI élevé illustre l'impact des politiques environnementales. Certains pays riches adoptent des stratégies agressives pour réduire leurs émissions, tandis que d'autres continuent à dépendre de sources d'énergie fossiles.

- **Équilibre entre prospérité et durabilité** :
    - Le graphique met en lumière le défi mondial consistant à atteindre un équilibre entre le développement humain et la préservation de l'environnement. Il souligne la nécessité pour les pays à HDI élevé de réduire leurs émissions tout en maintenant leur niveau de vie.

- **Rôle des ressources naturelles** :
    - Certains pays riches en ressources énergétiques (comme le pétrole ou le gaz) peuvent avoir des émissions élevées malgré des efforts de développement durable. Cela reflète les défis liés à la transition énergétique dans ces économies.

Impact des Sources d'énergie



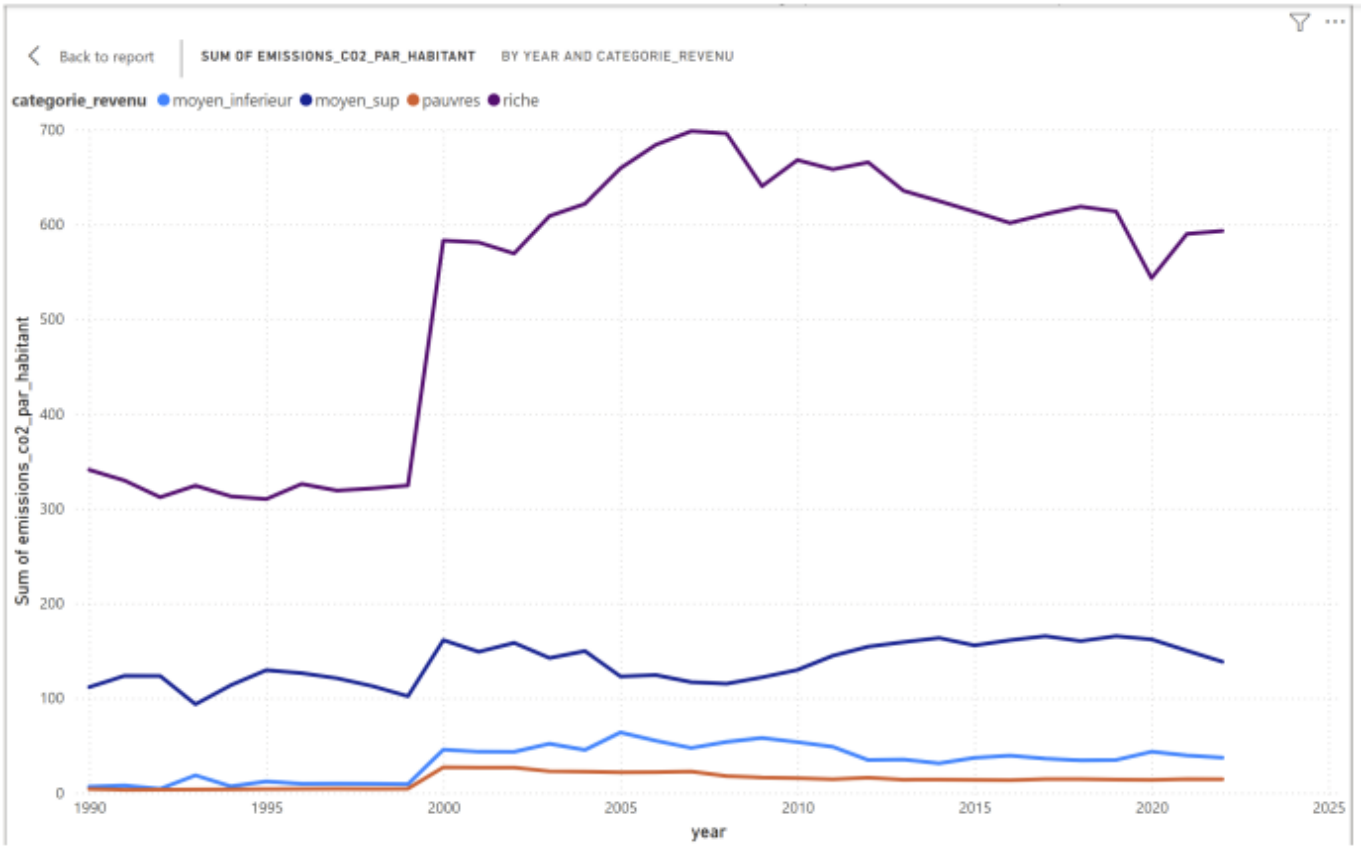
Observations

- **Cmbustibles fossiles** : Les combustibles fossiles représentent 77.56 % des émissions. Le charbon, le pétrole et le gaz naturel restent les principales sources d'énergie, malgré leurs impacts environnementaux négatifs.
- **Énergies Renouvelables** : Les énergies renouvelables (13.63 %) et nucléaires (3.48 %) contribuent pour une part minoritaire. Cela souligne le retard pris dans la transition énergétique mondiale.

Interprétation

- **Dépendance aux Fossiles** : La dépendance aux combustibles fossiles est un obstacle majeur à la réduction des émissions.
- **Urgence de la Transition** : Investir massivement dans les énergies renouvelables et nucléaires est essentiel pour réduire les émissions tout en satisfaisant la demande énergétique mondiale.

Évolution Temporelle des émissions de carbonne



Observations

- **Pays riches** : Les pays riches montrent une tendance à des émissions plus volatiles, avec un pic autour de 2010–2015. Cela pourrait être dû à des politiques de réduction des émissions mises en place après cette période.
- **Pays pauvres** : Les pays pauvres maintiennent des émissions stables et faibles, reflétant leur faible consommation d'énergie.

Interprétation

- **Efforts de Réduction** : Les pays riches semblent avoir commencé à réduire leurs émissions après 2015, probablement grâce à des politiques climatiques et des investissements dans les énergies propres.
- **Développement futur** : Les pays pauvres risquent d'accroître leurs émissions à mesure qu'ils se développent, sauf si des solutions durables leur sont proposées dès maintenant.

Conclusion Générale du Projet

Le projet que nous avons mené représente une chaîne complète d'intégration de données, allant de l'extraction des jeux de données bruts à leur transformation, puis à leur chargement dans un Data Warehouse relationnel. Cette approche structurée a permis de préparer les données pour une analyse décisionnelle avancée via Power BI.

Synthèse des Objectifs Atteints

1. **Extraction des données brutes** :

- Nous avons réussi à extraire des données socio-économiques provenant de sources fiables (Our World in Data), incluant :
  - L'indice de développement humain (HDI),
  - Les émissions de CO_2 par habitant,
  - Le PIB par habitant,
  - Les sources d'énergie principales utilisées par chaque pays.
- Ces données ont été chargées sous forme de DataFrames pandas pour être manipulées plus facilement.

2. **Transformation et nettoyage des données** :

- Une étape essentielle a consisté à nettoyer les valeurs manquantes, uniformiser les formats, et enrichir les jeux de données avec des catégories pertinentes :
  - Catégorisation de l'HDI selon les seuils définis (Faible, Moyen, Élevé, Très élevé).
  - Détermination de l'énergie principale utilisée par pays et année.
  - Classification des revenus selon le PIB par habitant.
  - Filtrage des émissions de CO_2 pour ne conserver que les années postérieures à 1990.
- À partir de ces données transformées, nous avons construit un **modèle étoile**, comprenant cinq tables de dimensions (`Dimension_Pays`, `Dimension_Temps`, `Dimension_Energy`, `Dimension_Developpement_Humain`, `Dimension_Revenus`) et une table de faits centrale (`Fait_SocioEconomique`).

3. **Chargement dans PostgreSQL** :

- Les données transformées ont été sauvegardées au format CSV, puis chargées dans une base de données PostgreSQL via le script `loader.py`.
- La création automatique des tables et la gestion des contraintes d'intégrité garantissent la cohérence des données dans le Data Warehouse.

4. **Visualisation via Power BI** :

- En connectant Power BI à la base PostgreSQL, nous avons pu explorer visuellement les tendances clés, notamment :
  - La corrélation entre le niveau de développement humain et les émissions de CO_2.
  - L'analyse comparative des stratégies énergétiques entre pays riches et pays pauvres.
  - L'évolution temporelle des indicateurs socio-économiques.

Réflexion sur les Résultats Obtenus

Les résultats obtenus montrent une forte disparité entre les pays en matière d'empreinte carbone et de développement économique :

- **Pays développés** : Bien que certains affichent un haut niveau de développement humain, leurs émissions de CO_2 restent élevées, souvent liées à leur dépendance aux énergies fossiles.
- **Pays en développement** : Malgré leurs faibles émissions, ils subissent davantage les effets du changement climatique, ce qui soulève des questions d'équité environnementale.

En outre, certaines exceptions notables montrent qu'il est possible d'allier un bon niveau de développement humain à des émissions réduites, grâce à des politiques énergétiques ambitieuses ou une transition vers les

énergies renouvelables.

Points Forts du Projet

- **Automatisation du pipeline ETL** : Le processus est entièrement automatisé, reproductible et facilement extensible.
- **Modélisation adaptée** : Le modèle étoile proposé permet une analyse multidimensionnelle efficace.
- **Exploitation décisionnelle** : Grâce à Power BI, les visualisations produites offrent des insights clairs et exploitables pour les décideurs.

Limites et Perspectives d'Amélioration

Malgré ses réussites, le projet présente quelques limites :

- **Étendue des données** : Certaines sources sont incomplètes ou manquent de données historiques profondes.
- **Performance du chargement** : Le chargement des fichiers CSV dans PostgreSQL peut être amélioré pour mieux gérer les volumes importants de données.
- **Couverture géographique** : Certains pays ne sont pas représentés dans toutes les sources, ce qui limite la généralisation des analyses.

Des perspectives futures pourraient inclure :

- **L'automatisation totale avec Docker et Airflow** pour orchestrer les étapes ETL.
- **L'intégration de nouvelles sources de données** pour enrichir l'analyse (données démographiques, données sur les politiques publiques, etc.).
- **La mise en place d'un serveur Power BI Service** pour partager les dashboards avec des utilisateurs finaux.

Conclusion Finale

Ce projet constitue une réponse concrète à un besoin croissant d'analyser les données socio-économiques pour mieux comprendre les dynamiques mondiales actuelles. En combinant les technologies Python, PostgreSQL et Power BI, il offre une solution robuste, modulaire et évolutive. Il démontre également comment la Business Intelligence peut contribuer à éclairer les choix politiques dans un contexte de développement durable et d'équité mondiale.