

Sprint Data – Projet Final

MAUFFRÉ Lukas – OHAYON Hillel – NAHRA Georgio – THIENOT Victor – DUHOO Lucas – HAMADI Aziz
Bachelor 2 – Groupe 3

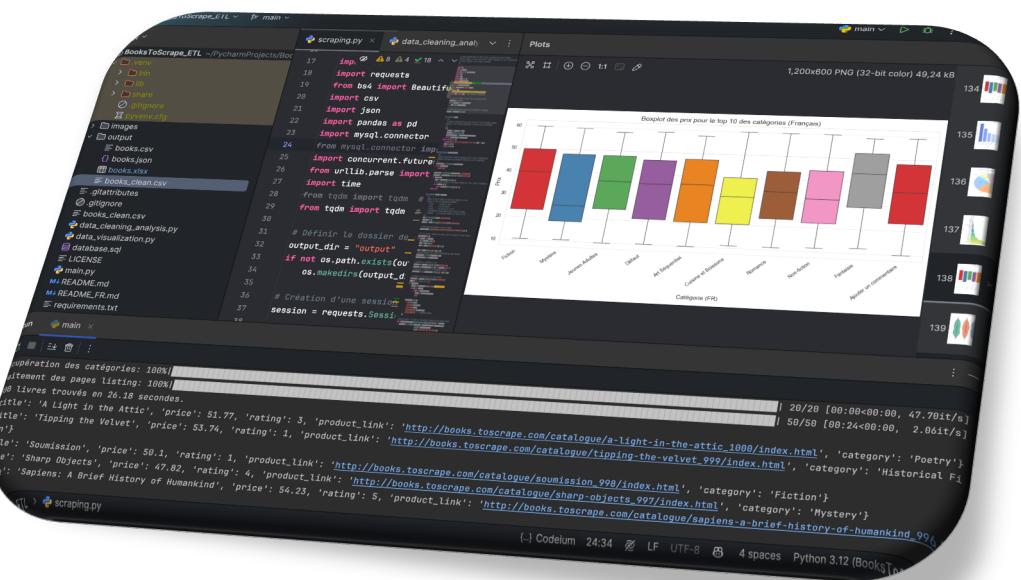


Table des matières

Acquis d'apprentissage.....	- 4 -
Objectifs	- 4 -
Note sur les extraits de code et les captures d'écran	- 4 -
1. <i>Introduction</i>	- 5 -
2. <i>Présentation des fichiers du projet</i>	- 5 -
3. <i>Scraping des données</i>	- 5 -
3.1. Objectif.....	- 5 -
3.2. Technologies utilisées	- 6 -
3.3. Extrait de code - Scraping.....	- 6 -
3.4. Données stockées	- 7 -
4. <i>Stockage en base de données MySQL</i>.....	- 8 -
4.1. Création de la base de données.....	- 8 -
4.2. Insertion des données dans la base.....	- 8 -
4.3. Vérification des données en base	- 9 -
5. <i>Nettoyage et Analyse des données</i>	- 10 -
5.1. Objectif.....	- 10 -
5.2. Opérations effectuées	- 10 -
5.3. Extrait de code - Nettoyage des données.....	- 10 -
5.4. Fichier généré après nettoyage.....	- 11 -
6. <i>Visualisation des données</i>	- 11 -
6.1. Objectif.....	- 11 -
6.2. Graphiques réalisés	- 11 -
6.3. Captures d'écran des visualisations	- 11 -
7. <i>Conclusion</i>	- 14 -
8. <i>Annexes</i>	- 14 -
8.1. Code Source Complet.....	- 14 -
8.1.1 Script de Scraping (scraping.py)	- 14 -
8.1.2 Script de Nettoyage et Analyse (data_cleaning_analysis.py).....	- 19 -
8.1.3 Script de Visualisation (data_visualization.py).....	- 24 -
8.1.4 Script Principal et Menu Interactif (main.py).....	- 30 -
8.1.5 Script SQL de la Base de Données (database.sql).....	- 32 -
8.2. Visualisations Supplémentaires.....	- 34 -
8.2.1 Distribution des livres par catégorie (Anglais et Français).....	- 34 -
8.2.2 Répartition des catégories sous forme de diagramme circulaire.....	- 34 -
8.2.3 Répartition des livres selon leur notation (rating)	- 35 -
8.2.4 Distribution des prix des livres	- 35 -
8.2.5 Répartition des livres par catégorie de prix	- 36 -
8.2.6 Boxplot des prix par rating	- 36 -
8.2.7 Boxplot des prix par catégorie (Anglais et Français)	- 37 -
8.2.8 Répartition de toutes les catégories (Anglais et Français).....	- 37 -
8.2.9 Violin plot des prix par rating	- 38 -
8.3. Explications Techniques Supplémentaires.....	- 38 -

8.3.1	Optimisation du Scraping avec ThreadPoolExecutor	- 38 -
8.3.2	Conversion et Nettoyage des Données avec Pandas.....	- 40 -
8.3.3	Explication des Visualisations	- 41 -
8.4.	Formats d'Exportation des Données.....	- 42 -
8.4.1	Export en CSV.....	- 42 -
8.4.2	Export en JSON.....	- 43 -
8.4.3	Export en Excel	- 44 -
8.5.	Exécution du Programme et Résultats en Console.....	- 44 -
8.6.	Conclusion de l'Annexe.....	- 48 -

Acquis d'apprentissage

À l'issue de ce projet, nous avons acquis et consolidé plusieurs compétences essentielles dans le domaine du traitement et de l'analyse de données :

- **Scraping Web** : Extraction automatisée de données depuis un site e-commerce en utilisant **BeautifulSoup** et **Requests**.
- **Manipulation de Bases de Données** : Création et gestion d'une base **MySQL**, requêtes SQL et intégration des données extraites.
- **Nettoyage et Préparation des Données** : Utilisation de **Pandas** pour traiter les données, gérer les valeurs manquantes, normaliser les catégories et enrichir les jeux de données.
- **Analyse et Visualisation** : Génération de graphiques pertinents à l'aide de **Matplotlib** et **Seaborn** pour extraire des insights exploitables.
- **Gestion de Projet et Structuration du Code** : Organisation d'un projet en plusieurs fichiers bien distincts (`scraping.py`, `data_cleaning_analysis.py`, `data_visualization.py`), respect des bonnes pratiques de développement.
- **Automatisation et Optimisation** : Utilisation de threading pour accélérer le scraping et structuration d'un pipeline de traitement des données en plusieurs étapes.
- **Stockage et Exportation** : Enregistrement des données sous **CSV**, **JSON** et **Excel** pour assurer une meilleure exploitation des résultats.
- **Rédaction Technique** : Capacité à documenter un projet technique en détaillant chaque étape avec des illustrations et des explications claires.

Objectifs

Ce projet avait pour but de mettre en pratique un ensemble de compétences en traitement et analyse de données en suivant une méthodologie complète. Les objectifs principaux étaient :

- **Extraire des données d'un site e-commerce** contenant des informations sur les livres (titre, prix, catégorie, notation, lien produit).
- **Stocker les données dans une base de données relationnelle (MySQL)** avec une structure bien définie.
- **Nettoyer et normaliser les données** pour les rendre exploitables en supprimant les erreurs, en convertissant les types et en enrichissant les colonnes existantes.
- **Réaliser des analyses descriptives et visualisations** pour identifier les tendances des livres disponibles (prix moyens, distribution des catégories, ratings).
- **Produire un pipeline ETL (Extract, Transform, Load)** bien organisé et réutilisable, avec des fichiers distincts pour chaque étape.
- **Automatiser le processus** grâce à un script interactif (`main.py`) permettant de lancer les différentes étapes du projet de manière intuitive.
- **Présenter et documenter le projet** de manière claire avec un rapport détaillé et structuré, incluant des illustrations des résultats obtenus.

Note sur les extraits de code et les captures d'écran

Les captures d'écran de code, de sorties et de visualisations présentées dans ce rapport sont **des extraits représentatifs** du projet.

Pour éviter d'alourdir le document, les versions complètes des scripts Python, du fichier SQL et des visualisations sont disponibles en [annexe](#).

Cette approche garantit une présentation **claire et concise**, tout en offrant un accès aux détails techniques pour une analyse approfondie.

1. Introduction

Ce projet s'inscrit dans le cadre du **SPRINT DATA** et a pour objectif d'extraire des données d'un site e-commerce, de les stocker dans une base de données **MySQL**, de les nettoyer avec **Pandas**, puis de les analyser et visualiser à l'aide de **Matplotlib** et **Seaborn**.

Le site sélectionné pour le scraping est **Books To Scrape**, un site fictif de vente de livres contenant des informations sur les produits (titre, prix, catégorie, notation, lien produit).

Le projet est structuré en plusieurs étapes :

- **Scraping des données**
- **Stockage en base de données MySQL**
- **Nettoyage et analyse des données**
- **Visualisation des résultats**

L'ensemble du projet a été codé en **Python**, et les données sont stockées dans une base **MySQL**. Les fichiers générés sont exportés aux formats **CSV**, **JSON** et **Excel**.

2. Présentation des fichiers du projet

Le projet est organisé de manière structurée en plusieurs fichiers :

Fichier	Rôle
scraping.py	Scraping des données du site Books To Scrape et stockage des résultats en fichiers CSV, JSON, Excel.
data_cleaning_analysis.py	Chargement, nettoyage et transformation des données avec Pandas .
data_visualization.py	Analyse et visualisation des données avec Matplotlib et Seaborn .
main.py	Menu interactif permettant d'exécuter les différentes étapes du projet.
database.sql	Script SQL pour la création de la base de données et la table books dans MySQL.

Les **fichiers générés** lors de l'exécution sont :

- **Données brutes** : books.csv, books.json, books.xlsx
 - **Données nettoyées** : books_clean.csv
 - **Visualisations (PNG)** : Graphiques d'analyse des données
-

3. Scraping des données

3.1. Objectif

L'objectif du scraping est d'extraire les informations des livres vendus sur le site **Books To Scrape**.

Les informations collectées pour chaque livre sont :

- **Titre**
- **Prix**
- **Note (rating)**
- **Lien vers la fiche produit**
- **Catégorie du livre**

3.2. Technologies utilisées

Le scraping est réalisé à l'aide de **BeautifulSoup** et **Requests**. L'approche utilisée permet d'extraire efficacement les informations de chaque page, en parcourant toutes les pages du catalogue.

3.3. Extrait de code - Scraping

L'extraction des données a été réalisée grâce à **BeautifulSoup** et **Requests**, en parcourant les pages du site **Books To Scrape**.

Chaque livre est analysé pour récupérer les informations essentielles : **titre, prix, notation, lien produit et catégorie**.

L'extrait de code ci-dessous montre la **fonction** `parse_books()`, qui analyse les pages et extrait les données pertinentes.

Ensuite, les informations collectées sont **sauvegardées dans plusieurs formats** (CSV, JSON, Excel), permettant une meilleure exploitation des données. Les fonctions `save_data_csv()`, `save_data_json()` et `save_data_excel()` assurent le stockage de ces données sous des formats adaptés à l'analyse et à la visualisation.

```
1 def parse_books(html, page_url):
2     """
3         Analyse le HTML et extrait les données des livres de la page.
4         Retourne une liste de dictionnaires contenant :
5             - title      : le titre du livre
6             - price      : le prix (float)
7             - rating     : la note (1 à 5)
8             - product_link : le lien absolu vers le détail du produit
9             - category   : la catégorie extraite de la page de détail
10        """
11    soup = BeautifulSoup(html, 'html.parser')
12    books_data = []
13    articles = soup.find_all('article', class_='product_pod')
14
15    # Dictionnaire pour convertir la note en entier
16    rating_mapping = {"One": 1, "Two": 2, "Three": 3, "Four": 4, "Five": 5}
17
18    for article in articles:
19        h3 = article.find('h3')
20        a_tag = h3.find('a')
21        title = a_tag.get('title', '').strip()
22
23        link = a_tag.get('href', '').strip()
24        product_link = urljoin(page_url, link)
25
26        price_tag = article.find('p', class_='price_color')
27        price_value = float(price_tag.text.replace('£', '')) if price_tag else 0.0
28
29        rating_tag = article.find('p', class_='star-rating')
30        rating = rating_mapping.get(rating_tag.get("class")[1], 0) if rating_tag else 0
31
32        books_data.append({
33            "title": title,
34            "price": price_value,
35            "rating": rating,
36            "product_link": product_link,
37            "category": None # La catégorie sera ajoutée après
38        })
39
40    return books_data
```

```

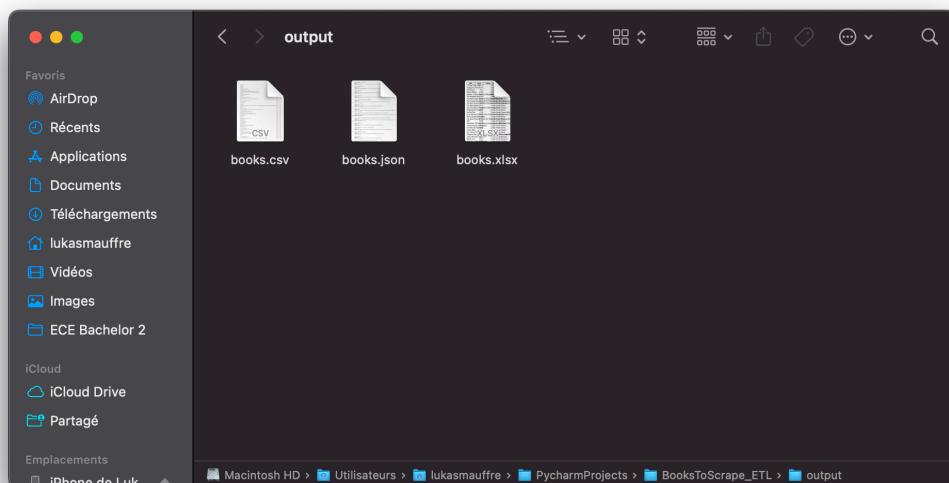
1  def save_data_csv(books, filename="books.csv"):
2      """
3          Sauvegarde la liste des livres dans un fichier CSV.
4      """
5      if not books:
6          print("Aucun livre à sauvegarder en CSV.")
7          return
8      keys = books[0].keys()
9      filepath = os.path.join(output_dir, filename)
10     with open(filepath, mode='w', newline='', encoding='utf-8') as f:
11         writer = csv.DictWriter(f, fieldnames=keys)
12         writer.writeheader()
13         writer.writerows(books)
14     print(f"Les données ont été sauvegardées dans {filepath}")
15
16 def save_data_json(books, filename="books.json"):
17     """
18         Sauvegarde la liste des livres dans un fichier JSON.
19     """
20     filepath = os.path.join(output_dir, filename)
21     with open(filepath, mode='w', encoding='utf-8') as f:
22         json.dump(books, f, ensure_ascii=False, indent=4)
23     print(f"Les données ont été sauvegardées dans {filepath}")
24
25 def save_data_excel(books, filename="books.xlsx"):
26     """
27         Sauvegarde la liste des livres dans un fichier Excel.
28     """
29     filepath = os.path.join(output_dir, filename)
30     df = pd.DataFrame(books)
31     df.to_excel(filepath, index=False)
32     print(f"Les données ont été sauvegardées dans {filepath}")

```

3.4. Données stockées

Les données collectées sont exportées sous **trois formats** :

- **CSV** : books.csv
- **JSON** : books.json
- **Excel** : books.xlsx



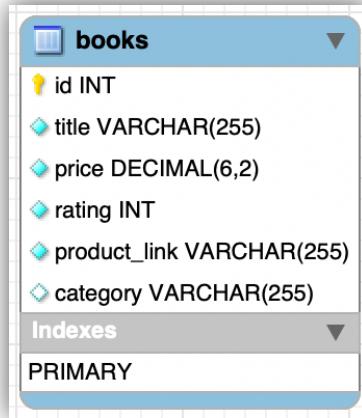
4. Stockage en base de données MySQL

4.1. Création de la base de données

Les données extraites sont insérées dans une base de données **MySQL**.

La base de données est créée avec la structure suivante :

Schéma de la table books



```
CREATE TABLE books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    price DECIMAL(6,2) NOT NULL,
    rating INT NOT NULL,
    product_link VARCHAR(255) NOT NULL,
    category VARCHAR(255) DEFAULT NULL
);
```

4.2. Insertion des données dans la base

Les données sont insérées en base grâce à la fonction `insert_data_mysql()` du fichier `scraping.py`.

Une fois les données extraites et sauvegardées sous différents formats (CSV, JSON, Excel), elles sont **insérées dans une base de données MySQL** pour permettre **une manipulation et une interrogation plus efficaces**.

L'insertion est réalisée grâce à la fonction `insert_data_mysql()` dans le fichier `scraping.py`, qui utilise **MySQL Connector** pour établir la connexion avec la base et insérer les données dans la table `books`.

Avant l'insertion, la base de données est créée avec la **commande CREATE DATABASE**, et la table `books` est définie en respectant une structure relationnelle avec une **clé primaire auto-incrémentée** (`id`) et des **contraintes de type** pour garantir l'intégrité des données.

L'extrait de code ci-dessous montre **les instructions SQL utilisées** pour créer la base et la table, insérer les données collectées, et effectuer des vérifications pour s'assurer de la bonne insertion des enregistrements.

```

1  -- Création de la base de données avec gestion des caractères Unicode
2  CREATE DATABASE IF NOT EXISTS books_scrape
3      DEFAULT CHARACTER SET utf8mb4
4      COLLATE utf8mb4_unicode_ci;
5
6  -- Sélection de la base de données
7  USE books_scrape;
8
9  -- Création de la table "books"
10 CREATE TABLE IF NOT EXISTS books (
11     id INT AUTO_INCREMENT PRIMARY KEY,
12     title VARCHAR(255) NOT NULL,
13     price DECIMAL(6,2) NOT NULL,
14     rating INT NOT NULL,
15     product_link VARCHAR(255) NOT NULL,
16     category VARCHAR(255) DEFAULT NULL
17 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
18
19 -- Insertion des données dans la table books
20 INSERT INTO books (title, price, rating, product_link, category)
21 VALUES
22     ('The Great Gatsby', 12.99, 5, 'http://books.toscrape.com/the-great-gatsby', 'Classics'),
23     ('To Kill a Mockingbird', 8.99, 4, 'http://books.toscrape.com/to-kill-a-mockingbird', 'Fiction'),
24     ('1984', 10.50, 5, 'http://books.toscrape.com/1984', 'Science Fiction'),
25     ('Moby-Dick', 15.00, 3, 'http://books.toscrape.com/moby-dick', 'Classics'),
26     ('Pride and Prejudice', 7.99, 5, 'http://books.toscrape.com/pride-and-prejudice', 'Romance');
27
28 -- Vérification de la structure et du contenu de la table books
29 SHOW TABLES;
30 DESCRIBE books;
31 SELECT * FROM books LIMIT 5;

```

4.3. Vérification des données en base

Les données peuvent être consultées avec la requête suivante :

```
SELECT * FROM books LIMIT 10;
```

			* id	* title	* price	* rating	* product_link	category
			int	varchar(255)	decimal(6,2)	int	varchar(255)	varchar(255)
1	>	1		A Light in the Attic	51.77	3	http://books.toscrape.com/cat	Poetry
2	>	2		Tipping the Velvet	53.74	1	http://books.toscrape.com/cat	Historical Fiction
3	>	3		Soumission	50.10	1	http://books.toscrape.com/cat	Fiction
4	>	4		Sharp Objects	47.82	4	http://books.toscrape.com/cat	Mystery
5	>	5		Sapiens: A Brief History of Humankind	54.23	5	http://books.toscrape.com/cat	History
6	>	6		The Requiem Red	22.65	1	http://books.toscrape.com/cat	Young Adult
7	>	7		The Dirty Little Secrets of Getting Rich	33.34	4	http://books.toscrape.com/cat	Business
8	>	8		The Coming Woman: A Novel in Three Parts	17.93	3	http://books.toscrape.com/cat	Default
9	>	9		The Boys in the Boat: Nine Americans and Their Dream at the 1936 Berlin Olympics	22.60	4	http://books.toscrape.com/cat	Default
10	>	10		The Black Maria	52.15	1	http://books.toscrape.com/cat	Poetry

5. Nettoyage et Analyse des données

5.1. Objectif

Les données brutes peuvent contenir des **valeurs manquantes**, des **types de données incorrects** ou des **incohérences**. L'objectif est de les nettoyer et de les enrichir avec des colonnes supplémentaires.

5.2. Opérations effectuées

Le script `data_cleaning_analysis.py` effectue les opérations suivantes :

- **Gestion des valeurs manquantes** (remplacement par défaut)
- **Conversion des types de données** (price en float, rating en int)
- **Ajout d'une colonne de catégorisation des prix** (Low, Medium, High)
- **Traduction des catégories de livres en français**

5.3. Extrait de code - Nettoyage des données

Après l'extraction et le stockage des données brutes, il est essentiel de **les nettoyer et les transformer** avant toute analyse.

Le fichier `data_cleaning_analysis.py` utilise **Pandas** pour appliquer différentes opérations de nettoyage :

- **Conversion des types de données** : transformation de la colonne price en **float** et de rating en **int**.
- **Gestion des valeurs manquantes** : remplacement des champs vides par des valeurs par défaut pour éviter les erreurs d'analyse.
- **Normalisation des catégories** : suppression des caractères parasites et uniformisation du texte.

L'extrait de code ci-dessous illustre **la conversion des prix en nombre flottant** (float) et **la gestion des valeurs manquantes**, permettant d'assurer la cohérence des données avant leur exploitation.

```
1 def clean_data(df):
2     """Nettoyage et préparation des données."""
3     print("Premières lignes des données brutes:")
4     print(df.head())
5
6     # Affichage des valeurs manquantes initiales
7     print("\nValeurs manquantes par colonne AVANT nettoyage:")
8     print(df.isnull().sum())
9
10    # Gestion des valeurs manquantes
11    df['title'] = df['title'].fillna("inconnu")
12    df['price'] = df['price'].fillna(0)
13    df['rating'] = df['rating'].fillna(0)
14    df['product_link'] = df['product_link'].fillna("inconnu")
15    df['category'] = df['category'].fillna("inconnu")
16
17    # Conversion du prix en nombre (float) : suppression du symbole '€' si présent
18    if df['price'].dtype == object:
19        df['price'] = df['price'].replace({'€': ''}, regex=True)
20    df['price'] = pd.to_numeric(df['price'], errors='coerce').fillna(0)
21
22    # Conversion du rating en entier
23    df['rating'] = pd.to_numeric(df['rating'], errors='coerce').fillna(0).astype(int)
24
25    # Affichage final des valeurs manquantes et des statistiques
26    print("\nValeurs manquantes par colonne APRÈS nettoyage:")
27    print(df.isnull().sum())
28    print("\nStatistiques descriptives:")
29    print(df.describe(include='all'))
30
31    return df
```

5.4. Fichier généré après nettoyage

- books_clean.csv (données nettoyées et enrichies)

Après l'étape de nettoyage, les données sont **enrichies et transformées** pour être plus facilement exploitable.

Le fichier books_clean.csv contient désormais :

- **Des valeurs sans erreurs** : toutes les valeurs manquantes ont été remplacées.
- **Des types de données cohérents** : price est converti en float, rating en int.
- **De nouvelles colonnes** comme price_category, qui classe les livres selon leur prix.

L'image ci-dessous montre un aperçu du fichier books_clean.csv, où l'on peut observer les données nettoyées et prêtes à être analysées.

	title_normalized	category_normalized	category_fr	price_category	rating_j
1	a light in the attic	poetry	Poésie	High	
2	tipping the velvet	historical fiction	Fiction Historique	High	
3	soumission	fiction	Fiction	High	
4	sharp objects	mystery	Mystère	Medium	
5	sapiens: a brief history of humankind	history	Histoire	High	
6	the requiem red	young adult	Jeunes Adultes	Medium	
7	the dirty little secrets of getting your dream ...	business	Affaires	Medium	
8	the coming woman: a novel based on the life of ...	default	Défaut	Low	
9	the boys in the boat: nine americans and their ...	default	Défaut	Medium	
10	the black maria	poetry	Poésie	High	

6. Visualisation des données

6.1. Objectif

L'objectif de cette étape est d'analyser les données grâce à des **visualisations graphiques** et d'extraire des insights pertinents.

6.2. Graphiques réalisés

Les graphiques ont été générés avec **Matplotlib et Seaborn** à partir du fichier data_visualization.py :

- **Distribution des livres par rating (countplot)**
- **Distribution des livres par catégorie de prix (barplot)**
- **Histogramme des prix avec courbe KDE (histplot)**
- **Boxplot des prix par rating**
- **Top 10 des catégories de livres (anglais et français) (barplot)**
- **Diagramme circulaire des principales catégories (pie chart)**

6.3. Captures d'écran des visualisations

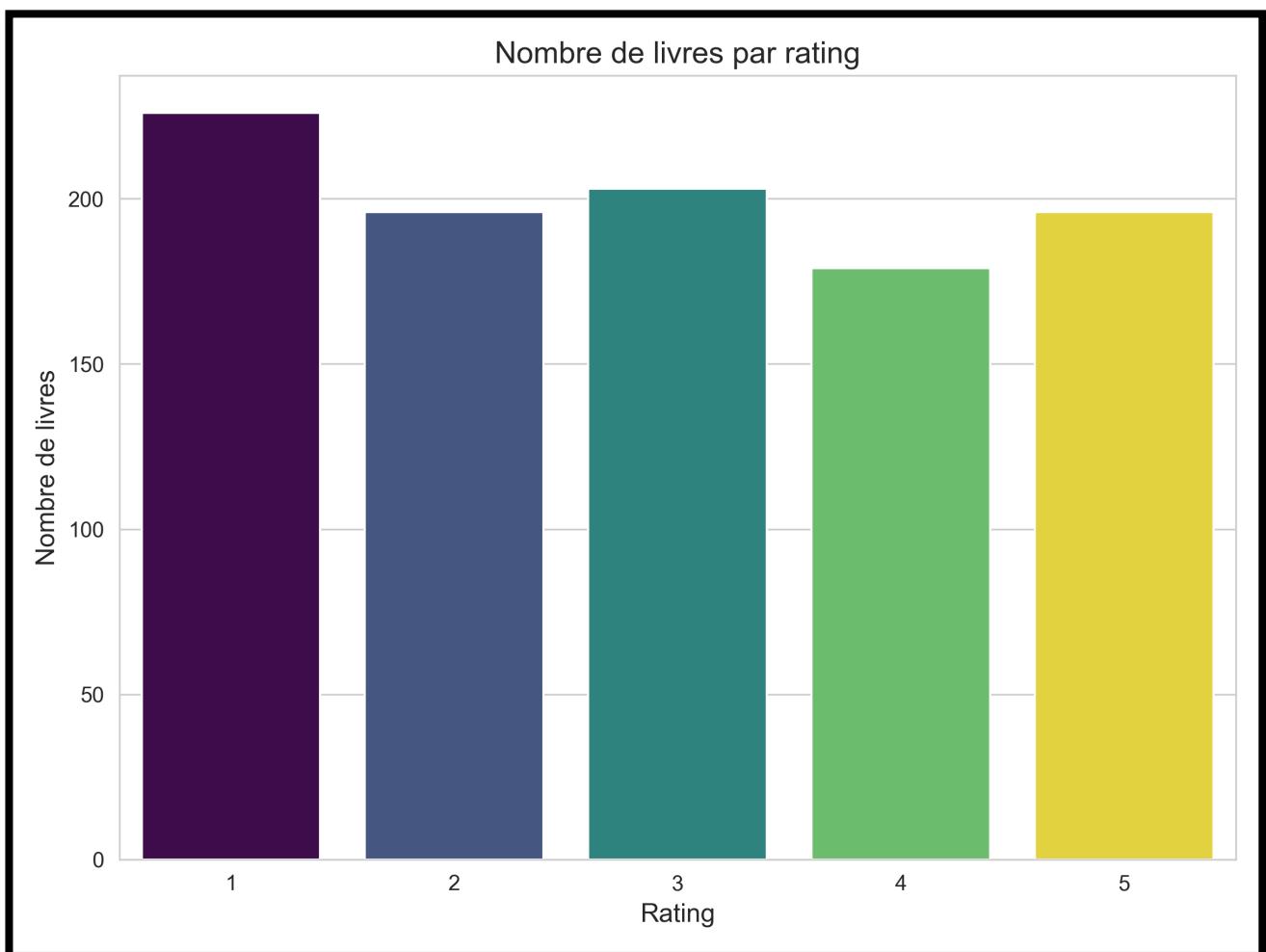
Une fois les données nettoyées, elles sont analysées à l'aide de **Seaborn** et **Matplotlib** pour extraire des tendances intéressantes.

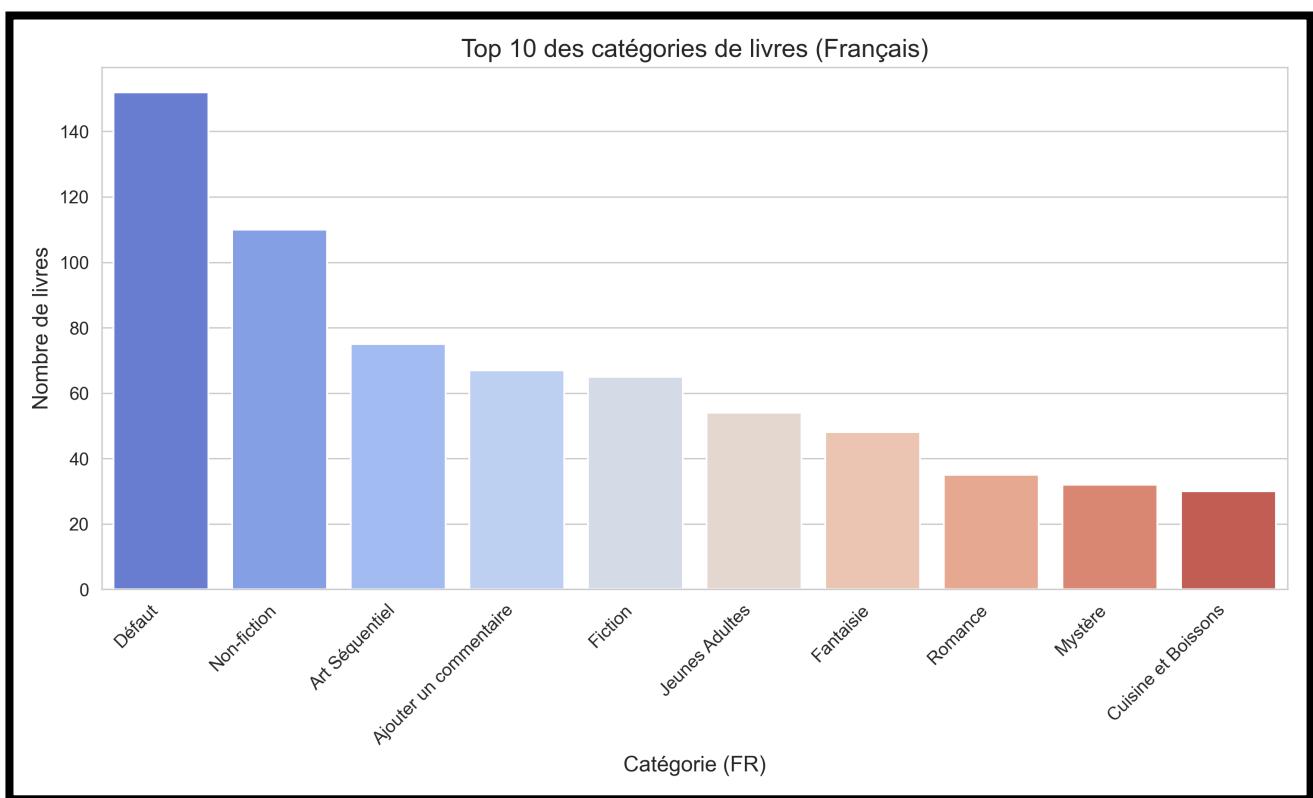
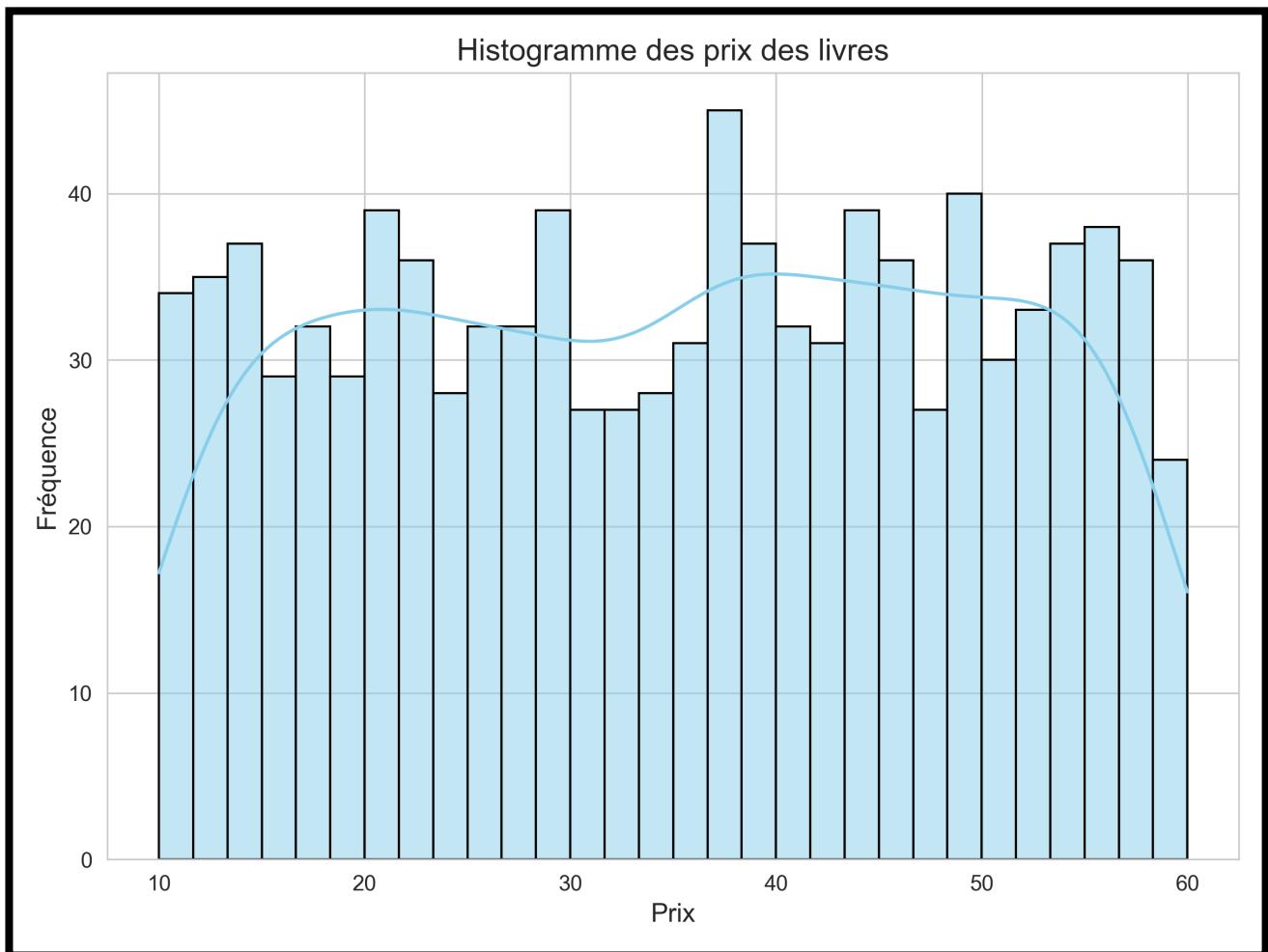
Les graphiques permettent de mieux comprendre **la distribution des prix, la répartition des catégories, et l'impact des notations sur les prix**.

Parmi les nombreuses visualisations générées, voici les plus représentatives :

1. **Distribution des livres par rating** → Permet d'identifier la répartition des notes attribuées aux livres.
2. **Histogramme des prix avec courbe KDE** → Donne une vision globale de la répartition des prix des livres.
3. **Top 10 des catégories de livres (en français)** → Montre les genres les plus représentés dans le dataset.

Les autres graphiques, tels que **les boxplots, pie charts et violin plots**, sont disponibles en **annexe**.





7. Conclusion

Ce projet a permis de **scraper, stocker, nettoyer et visualiser** des données provenant d'un site e-commerce en utilisant un pipeline de traitement **ETL (Extract, Transform, Load)**.

Les différentes étapes ont permis de :

- Extraire des données de Books To Scrape grâce à BeautifulSoup.**
- Stocker les données dans une base de données MySQL.**
- Nettoyer et enrichir les données avec Pandas.**
- Créer des visualisations pertinentes pour analyser les tendances.**

Améliorations possibles :

- Ajouter d'autres sources de données pour un dataset plus riche.
- Mettre en place un dashboard interactif pour l'exploration des données.
- Automatiser la mise à jour des données en temps réel.

8. Annexes

ANNEXES – Détails Techniques du Projet

L'annexe regroupe **toutes les versions complètes des codes sources, les visualisations supplémentaires**, ainsi que **des précisions techniques** sur certains aspects du projet. Elle permet d'approfondir chaque étape sans surcharger le rapport principal.

8.1. Code Source Complet

Les scripts suivants ont été utilisés pour la réalisation du projet. Chaque section détaille le rôle du script et fournit son contenu intégral.

8.1.1 Script de Scraping (scraping.py)

Le fichier `scraping.py` est le premier élément du pipeline ETL de ce projet. Il est chargé d'extraire les informations des livres disponibles sur le site **Books To Scrape**.

Le script suit plusieurs étapes essentielles pour assurer **une récupération efficace et structurée des données** :

1. **Téléchargement du contenu HTML** de chaque page du site en gérant les erreurs réseau.
2. **Extraction des informations des livres** : titre, prix, notation, lien produit et catégorie.
3. **Optimisation avec ThreadPoolExecutor** pour récupérer les catégories en parallèle et accélérer le scraping.
4. **Parcours automatique de la pagination** pour extraire tous les livres disponibles.
5. **Stockage des données** dans différents formats (**CSV, JSON, Excel**).
6. **Insertion des données dans une base MySQL** pour faciliter leur exploitation et analyse.

Bibliothèques utilisées

Le script repose sur plusieurs bibliothèques Python qui assurent ses différentes fonctionnalités :

- **requests** → Effectue les requêtes HTTP pour récupérer les pages du site.
- **BeautifulSoup (bs4)** → Analyse le HTML pour extraire les informations des livres.

- **csv, json, pandas** → Permettent de sauvegarder les données extraites sous différents formats (CSV, JSON, Excel).
- **mysql.connector** → Gère la connexion et l'insertion des données dans MySQL.
- **concurrent.futures.ThreadPoolExecutor** → Accélère le scraping en récupérant plusieurs pages et catégories en parallèle.
- **urllib.parse.urljoin** → Convertit les liens relatifs des produits en liens absous.
- **time** → Gère les pauses entre les requêtes pour éviter de surcharger le site.
- **tqdm** → Affiche une barre de progression lors de l'extraction des données.
- **os** → Vérifie et crée les dossiers nécessaires pour stocker les fichiers générés.

L'approche multi-threading avec **ThreadPoolExecutor** permet **d'optimiser le temps d'extraction** en effectuant plusieurs requêtes simultanément.

Grâce à BeautifulSoup, le script est capable **de parcourir les pages HTML et d'extraire précisément les informations nécessaires**.

Enfin, l'intégration avec MySQL via **mysql.connector** assure **une gestion efficace et structurée des données collectées**.

 **Le code source intégral de scraping.py est disponible ci-dessous.**

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Script de scraping pour Books to Scrape avec insertion dans la base de données MySQL.
5  Ce script extrait pour chaque livre :
6      - Le titre
7      - Le prix
8      - La note (rating)
9      - Le lien vers la fiche produit
10     - La catégorie (ex. "Historical Fiction")
11
12 Le script parcourt toutes les pages du site et sauvegarde les données extraites
13 dans des fichiers CSV, JSON, Excel (dans un dossier 'output'), puis insère ces données dans la base de données MySQL.
14 Il affiche également la durée totale du scraping.
15 """
16
17 import os
18 import requests
19 from bs4 import BeautifulSoup
20 import csv
21 import json
22 import pandas as pd
23 import mysql.connector
24 from mysql.connector import Error
25 import concurrent.futures
26 from urllib.parse import urljoin
27 import time
28 from tqdm import tqdm # pour afficher la progression
29 from tqdm import tqdm # Pour afficher la progression
30
31 # Définir le dossier de sortie pour les fichiers exportés
32 output_dir = "output"
33 if not os.path.exists(output_dir):
34     os.makedirs(output_dir)
35
36 # Création d'une session globale pour réutiliser les connexions
37 session = requests.Session()
38
39 def fetch_page(url):
40     """
41     Télécharge le contenu HTML de la page spécifiée en gérant l'encodage Unicode.
42     Retourne un tuple (html, effective_url) ou (None, url) en cas d'erreur.
43     """
44     try:
45         response = session.get(url)
46         response.raise_for_status()
47         response.encoding = response.apparent_encoding
48         return response.text, response.url
49     except Exception as e:
50         print(f"Erreur lors du téléchargement de la page {url} : {e}")
51         time.sleep(0.2)
52     return None, url

```

```

53
54 def fetch_category(product_link):
55     """
56     Récupère la catégorie du livre en analysant la page détaillée (product_link).
57     Retourne la catégorie (ex. 'Historical Fiction') ou None si introuvable.
58     """
59     detail_html, _ = fetch_page(product_link)
60     if not detail_html:
61         return None
62
63     soup_detail = BeautifulSoup(detail_html, "html.parser")
64     breadcrumb = soup_detail.find("ul", class_="breadcrumb")
65     if breadcrumb:
66         links = breadcrumb.find_all("a")
67         # Sur Books To Scrape, le fil d'ariane typique est : Home > Books > [Category] > [Titre sans lien]
68         if len(links) >= 3:
69             return links[-1].get_text(strip=True)
70     return None
71
72 def parse_books(html, page_url):
73     """
74     Analyse le HTML et extrait les données des livres de la page.
75     Retourne une liste de dictionnaires contenant :
76         - title      : le titre du livre
77         - price      : le prix (float)
78         - rating     : la note (1, 2, 3, 4, 5)
79         - product_link : le lien absolu vers le détail du produit
80         - category   : la catégorie extraite de la page de détail
81     """
82     soup = BeautifulSoup(html, 'html.parser')
83     books_data = []
84     articles = soup.find_all('article', class_='product_pod')
85
86     # Dictionnaire pour convertir la note en entier
87     rating_mapping = {"One": 1, "Two": 2, "Three": 3, "Four": 4, "Five": 5}
88
89     # Première boucle : extraire les infos de base et collecter les liens produits
90     product_links = []
91     for article in articles:
92         h3 = article.find('h3')
93         a_tag = h3.find('a')
94         title = a_tag.get('title', '').strip()
95
96         link = a_tag.get('href', '').strip()
97         # Utiliser page_url pour résoudre correctement les liens relatifs
98         product_link = urljoin(page_url, link)
99         product_links.append(product_link)
100
101        # Extraction et conversion du prix
102        price_tag = article.find('p', class_='price_color')
103        price_str = price_tag.text.strip() if price_tag else "0"
104        # Suppression du symbole '£' et conversion en float
105        price_value = float(price_str.replace('£', '')).strip()
106
107        # Extraction et conversion de la note (rating)
108        rating_tag = article.find('p', class_='star-rating')
109        rating = 0 # valeur par défaut
110        if rating_tag:
111            classes = rating_tag.get("class", [])
112            for cls in classes:
113                if cls in rating_mapping:
114                    rating = rating_mapping[cls]
115                    break
116
117            books_data.append({
118                "title": title,
119                "price": price_value,
120                "rating": rating,
121                "product_link": product_link,
122                "category": None # à renseigner ensuite
123            })
124
125        # Deuxième boucle : récupérer les catégories en parallèle pour chaque produit
126        with concurrent.futures.ThreadPoolExecutor(max_workers=15) as executor:
127            categories = list(tqdm(executor.map(fetch_category, product_links),
128                                    total=len(product_links),
129                                    desc="Récupération des catégories"))
130
131        # Affecter les catégories aux livres
132        for i, book in enumerate(books_data):
133            book["category"] = categories[i]
134
135    return books_data

```

```

135
136     def fetch_all_pages(base_url):
137         """
138             Parcourt toutes les pages du site en suivant la pagination.
139             Retourne la liste complète des livres extraits.
140         """
141         current_url = base_url
142         all_books = []
143         while current_url:
144             print("Scraping page :", current_url)
145             html, effective_url = fetch_page(current_url)
146             if html is None:
147                 break
148             books = parse_books(html, effective_url)
149             all_books.extend(books)
150             soup = BeautifulSoup(html, 'html.parser')
151             next_li = soup.find('li', class_='next')
152             if next_li:
153                 next_a = next_li.find('a')
154                 if next_a:
155                     next_href = next_a.get('href')
156                     current_url = urljoin(effective_url, next_href)
157                 else:
158                     break
159             else:
160                 break
161         return all_books
162
163     def fetch_all_pages_concurrent(base_url):
164         """
165             Parcourt toutes les pages du site en parallèle en suivant la pagination.
166             On part du principe que le site comporte 50 pages (ou on peut extraire ce nombre depuis la première page).
167             Retourne la liste complète des livres extraits.
168         """
169         first_html, first_effective_url = fetch_page(base_url)
170         if not first_html:
171             return []
172         soup = BeautifulSoup(first_html, 'html.parser')
173         current_page_text = soup.find("li", class_="current")
174         if current_page_text:
175             try:
176                 num_pages = int(current_page_text.get_text(strip=True).split()[-1])
177             except Exception as e:
178                 print("Impossible d'extraire le nombre de pages, utilisation de 50 par défaut.")
179                 num_pages = 50
180         else:
181             num_pages = 50
182
183         # Générer la liste des URLs : la première page est base_url, puis les pages suivantes
184         urls = [base_url]
185         for i in range(2, num_pages + 1):
186             page_url = f"http://books.toscrape.com/catalogue/page-{i}.html"
187             urls.append(page_url)
188
189         books = []
190         with concurrent.futures.ThreadPoolExecutor(max_workers=20) as executor:
191             results = list(executor.map(fetch_page, urls))
192             # Utilisation de tqdm pour afficher la progression dans le traitement des pages
193             for html, effective_url in tqdm(results, total=len(results), desc="Traitement des pages listing"):
194                 if html:
195                     books.extend(parse_books(html, effective_url))
196
197         return books

```

```

197
198     def save_data_csv(books, filename="books.csv"):
199         """
200             Sauvegarde la liste des livres dans un fichier CSV dans le dossier 'output'.
201         """
202         if not books:
203             print("Aucun livre à sauvegarder en CSV.")
204             return
205         keys = books[0].keys()
206         try:
207             filepath = os.path.join(output_dir, filename)
208             with open(filepath, mode='w', newline='', encoding='utf-8') as f:
209                 writer = csv.DictWriter(f, fieldnames=keys)
210                 writer.writeheader()
211                 for book in books:
212                     writer.writerow(book)
213             print(f"Les données ont été sauvegardées dans le fichier {filepath}")
214         except Exception as e:
215             print(f"Erreur lors de la sauvegarde en CSV : {e}")
216
217     def save_data_json(books, filename="books.json"):
218         """
219             Sauvegarde la liste des livres dans un fichier JSON dans le dossier 'output'.
220         """
221         try:
222             filepath = os.path.join(output_dir, filename)
223             with open(filepath, mode='w', encoding='utf-8') as f:
224                 json.dump(books, f, ensure_ascii=False, indent=4)
225             print(f"Les données ont été sauvegardées dans le fichier {filepath}")
226         except Exception as e:
227             print(f"Erreur lors de la sauvegarde en JSON : {e}")
228
229     def save_data_excel(books, filename="books.xlsx"):
230         """
231             Sauvegarde la liste des livres dans un fichier Excel dans le dossier 'output'.
232         """
233         try:
234             filepath = os.path.join(output_dir, filename)
235             df = pd.DataFrame(books)
236             df.to_excel(filepath, index=False)
237             print(f"Les données ont été sauvegardées dans le fichier {filepath}")
238         except Exception as e:
239             print(f"Erreur lors de la sauvegarde en Excel : {e}")

```

```

240
241 def insert_data_mysql(books, host='localhost', user='root', password='12345678', database='books_scrape'):
242     """
243     Insère la liste des livres dans la table MySQL après avoir vidé la table.
244     AVANT d'exécuter ce script, assurez-vous que la table 'books'
245     contient bien une colonne 'category' (ex. ALTER TABLE books ADD COLUMN category VARCHAR(255));
246     """
247     connection = None
248     try:
249         connection = mysql.connector.connect(
250             host=host,
251             user=user,
252             password=password,
253             database=database,
254             auth_plugin='mysql_native_password'
255         )
256         if connection.is_connected():
257             cursor = connection.cursor()
258             cursor.execute("TRUNCATE TABLE books;")
259             connection.commit()
260
261         insert_query = """
262             INSERT INTO books (title, price, rating, product_link, category)
263             VALUES (%s, %s, %s, %s, %s)
264             """
265         for book in books:
266             data = (
267                 book['title'],
268                 book['price'],
269                 book['rating'],
270                 book['product_link'],
271                 book['category']
272             )
273             cursor.execute(insert_query, data)
274             connection.commit()
275             print("Les données ont été insérées dans la base de données MySQL après vidage de la table.")
276     except mysql.connector.Error as e:
277         print("Erreur lors de l'insertion dans MySQL :", e)
278     finally:
279         if connection is not None and connection.is_connected():
280             cursor.close()
281             connection.close()
282
283 def main():
284     base_url = "http://books.toscrape.com/"
285     print("Début du scraping du site :", base_url)
286     start_time = time.time() # Démarrage du chronomètre
287
288     books = fetch_all_pages_concurrent(base_url)
289     duration = time.time() - start_time # Durée totale du scraping
290     print(f"{len(books)} livres trouvés en {duration:.2f} secondes.")
291
292     # Affichage de quelques livres pour vérification
293     for book in books[:5]:
294         print(book)
295
296     # Sauvegarde des données dans différents formats dans le dossier 'output'
297     save_data_csv(books)
298     save_data_json(books)
299     save_data_excel(books)
300
301     # Insertion des données dans la base de données MySQL
302     insert_data_mysql(books)
303
304 if __name__ == "__main__":
305     main()

```

8.1.2 Script de Nettoyage et Analyse (data_cleaning_analysis.py)

Le fichier **data_cleaning_analysis.py** est une étape clé du pipeline ETL. Après l'extraction des données brutes, ce script applique **différentes transformations et corrections** pour garantir leur qualité et leur exploitabilité.

Principales étapes du nettoyage et de l'analyse :

1. **Chargement des données** depuis le fichier CSV contenant les informations brutes issues du scraping.
2. **Gestion des valeurs manquantes** en remplaçant les champs vides par des valeurs par défaut (ex. prix à 0, titre inconnu).
3. **Conversion des types de données** :
 - **Prix** transformé en float, avec suppression du symbole €.
 - **Notation** convertie en int pour faciliter l'analyse.
4. **Normalisation des textes** (ex. conversion en minuscules, suppression des accents) pour assurer la cohérence des catégories.
5. **Traduction des catégories** : chaque catégorie en anglais est **convertie en français** pour améliorer la lisibilité.
6. **Ajout de colonnes calculées** :
 - **price_category** → Classe les livres en catégories de prix : Low (<20€), Medium (20-50€), High (>50€).
 - **category_fr** → Version française de la catégorie du livre.
7. **Analyse exploratoire des données** :
 - Comptage des livres par **notation (rating)**.
 - Moyenne des prix par **niveau de notation**.
 - Répartition des livres par **catégorie** et par **tranche de prix**.
8. **Sauvegarde des données nettoyées** dans un nouveau fichier CSV (books_clean.csv), prêt à être analysé ou visualisé.

Bibliothèques utilisées

Le script repose sur plusieurs bibliothèques Python essentielles pour le traitement des données :

- **pandas** → Manipulation des données et conversion des types.
- **numpy** → Gestion des valeurs numériques, notamment la catégorisation des prix.
- **unicodedata** → Suppression des accents et normalisation du texte.
- **os** → Gestion des fichiers et dossiers pour l'import/export des données.

Grâce à pandas, il est possible **d'automatiser le nettoyage des données brutes**, garantissant ainsi une cohérence et une meilleure qualité avant l'analyse.

L'utilisation de numpy permet **de segmenter automatiquement les prix** en catégories distinctes, facilitant ainsi les futures visualisations.

 **Le code source intégral de data_cleaning_analysis.py est disponible ci-dessous.**

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Script de nettoyage et d'analyse des données pour Books to Scrape.
5  Ce script charge les données à partir du fichier CSV (situé dans le dossier 'output'),
6  nettoie et prépare les données en effectuant :
7      - Conversion des types de données
8      - Gestion des valeurs manquantes
9      - Ajout de colonnes calculées pertinentes
10     - Normalisation des catégories (textes et colonnes catégorielles)
11     - Traduction des catégories en français
12 Les données nettoyées sont ensuite sauvegardées dans un nouveau fichier CSV dans le dossier 'output'.
13 """
14
15 import os
16 import pandas as pd
17 import numpy as np
18 import unicodedata
19
20 def load_data(filename=None):
21     """Charge les données à partir du fichier CSV situé dans le dossier 'output'. """
22     if filename is None:
23         filename = os.path.join("output", "books.csv")
24     try:
25         df = pd.read_csv(filename, encoding='utf-8')
26         print("Données chargées avec succès depuis", filename)
27         return df
28     except Exception as e:
29         print("Erreur lors du chargement des données :", e)
30     return None
31
32 def normalize_text(text):
33     """Normalise le texte : supprime les accents et convertit en minuscules. """
34     if isinstance(text, str):
35         # Supprimer les accents
36         text = unicodedata.normalize('NFKD', text).encode('ASCII', 'ignore').decode('utf-8')
37         # Convertir en minuscules
38         return text.lower().strip()
39     return text

```

```

40
41     def translate_category(cat):
42         """
43             Traduit la catégorie en anglais en français.
44             Si la catégorie n'est pas dans le dictionnaire, elle est retournée telle quelle.
45         """
46         translations = {
47             "books": "Livres",
48             "travel": "Voyage",
49             "mystery": "Mystère",
50             "historical fiction": "Fiction Historique",
51             "sequential art": "Art Séquentiel",
52             "classics": "Classiques",
53             "philosophy": "Philosophie",
54             "romance": "Romance",
55             "womens fiction": "Fiction Féminine",
56             "fiction": "Fiction",
57             "childrens": "Pour Enfants",
58             "religion": "Religion",
59             "nonfiction": "Non-fiction",
60             "music": "Musique",
61             "default": "Défaut",
62             "science fiction": "Science-fiction",
63             "sports and games": "Sports et Jeux",
64             "add a comment": "Ajouter un commentaire",
65             "fantasy": "Fantaisie",
66             "new adult": "Nouveaux Adultes",
67             "young adult": "Jeunes Adultes",
68             "science": "Science",
69             "poetry": "Poésie",
70             "paranormal": "Paranormal",
71             "art": "Art",
72             "psychology": "Psychologie",
73             "autobiography": "Autobiographie",
74             "parenting": "Parentalité",
75             "adult fiction": "Fiction pour adultes",
76             "humor": "Humour",
77             "horror": "Horreur",
78             "history": "Histoire",
79             "food and drink": "Cuisine et Boissons",
80             "christian fiction": "Fiction Chrétienne",
81             "business": "Affaires",
82             "biography": "Biographie",
83             "thriller": "Thriller",
84             "contemporary": "Contemporain",
85             "spirituality": "Spiritualité",
86             "academic": "Académique",
87             "self help": "Développement Personnel",
88             "historical": "Historique",
89             "christian": "Chrétien",
90             "suspense": "Suspense",
91             "short stories": "Nouvelles",
92             "novels": "Romans",
93             "health": "Santé",
94             "politics": "Politique",
95             "cultural": "Culturel",
96             "erotica": "Érotisme",
97             "crime": "Crime"
98         }
99         cat_norm = cat.lower().strip()
100        return translations.get(cat_norm, cat)

```

```

101
102 def clean_data(df):
103     """Nettoyage et préparation des données."""
104     print("Premières lignes des données brutes:")
105     print(df.head())
106
107     # Affichage des valeurs manquantes initiales
108     print("\nValeurs manquantes par colonne AVANT nettoyage:")
109     print(df.isnull().sum())
110
111     # Gestion des valeurs manquantes
112     df['title'] = df['title'].fillna("inconnu")
113     df['price'] = df['price'].fillna(0)
114     df['rating'] = df['rating'].fillna(0)
115     df['product_link'] = df['product_link'].fillna("inconnu")
116     df['category'] = df['category'].fillna("inconnu")
117
118     # Conversion des types de données
119     # Conversion du prix en nombre (float) : suppression éventuelle du symbole '€'
120     if df['price'].dtype == object:
121         df['price'] = df['price'].replace({'€': ''}, regex=True)
122     df['price'] = pd.to_numeric(df['price'], errors='coerce').fillna(0)
123
124     # Conversion du rating en entier puis en catégorie
125     df['rating'] = pd.to_numeric(df['rating'], errors='coerce').fillna(0).astype(int)
126
127     # Normalisation du texte pour la colonne title et la colonne category
128     df['title_normalized'] = df['title'].apply(normalize_text)
129     df['category_normalized'] = df['category'].apply(normalize_text)
130
131     # Traduction des catégories en français et création d'une nouvelle colonne
132     df['category_fr'] = df['category_normalized'].apply(translate_category)
133
134     # Ajout d'une colonne calculée : catégorisation du prix
135     # Exemple de catégorisation : Low (<20), Medium (20-50), High (>50)
136     categories_price = ['Low', 'Medium', 'High']
137     df['price_category'] = pd.cut(df['price'], bins=[-np.inf, 20, 50, np.inf], labels=categories_price)
138
139     # Normalisation des colonnes catégorielles : conversion en type "category"
140     df['price_category'] = df['price_category'].astype('category')
141     df['rating'] = df['rating'].astype('category')
142     df['category'] = df['category_normalized'].astype('category')
143     df['category_fr'] = df['category_fr'].astype('category')
144
145     # Affichage final des valeurs manquantes et des statistiques
146     print("\nValeurs manquantes par colonne APRÈS nettoyage:")
147     print(df.isnull().sum())
148     print("\nStatistiques descriptives:")
149     print(df.describe(include='all'))
150
151     return df

```

```

152
153 def analyze_data(df):
154     """Réalise quelques analyses descriptives sur les données nettoyées."""
155     # Nombre de livres par rating
156     rating_counts = df['rating'].value_counts().sort_index()
157     print("\nNombre de livres par rating:")
158     print(rating_counts)
159
160     # Prix moyen par rating (attention : rating est une catégorie, on le convertit en int pour le calcul)
161     df['rating_int'] = df['rating'].astype(int)
162     avg_price_rating = df.groupby('rating_int')['price'].mean()
163     print("\nPrix moyen par rating:")
164     print(avg_price_rating)
165
166     # Nombre de livres par catégorie de prix
167     price_category_counts = df['price_category'].value_counts()
168     print("\nNombre de livres par catégorie de prix:")
169     print(price_category_counts)
170
171     # Nombre de livres par catégorie (normalisée)
172     category_counts = df['category'].value_counts()
173     print("\nNombre de livres par catégorie:")
174     print(category_counts)
175
176     category_fr_counts = df['category_fr'].value_counts()
177     print("\nNombre de livres par catégorie (en français):")
178     print(category_fr_counts)
179
180 def save_clean_data(df, filename=None):
181     """Sauvegarde les données nettoyées dans un fichier CSV situé dans le dossier 'output'."""
182     if filename is None:
183         filename = os.path.join("output", "books_clean.csv")
184     # Création du dossier de sortie s'il n'existe pas
185     output_dir = os.path.dirname(filename)
186     if not os.path.exists(output_dir):
187         os.makedirs(output_dir)
188     try:
189         df.to_csv(filename, index=False, encoding='utf-8')
190         print(f"\nLes données nettoyées ont été sauvegardées dans {filename}")
191     except Exception as e:
192         print("Erreur lors de la sauvegarde des données :", e)
193
194 def main():
195     df = load_data()
196     if df is None:
197         return
198
199     df_clean = clean_data(df)
200     analyze_data(df_clean)
201     save_clean_data(df_clean)
202
203 if __name__ == '__main__':
204     main()

```

8.1.3 Script de Visualisation (data_visualization.py)

Le fichier **data_visualization.py** est la dernière étape du pipeline ETL. Il permet de **mieux comprendre et interpréter les données extraites et nettoyées** en générant plusieurs types de graphiques.

Ces visualisations facilitent l'analyse des tendances et des structures des livres disponibles sur **Books To Scrape**.

Principales étapes de la visualisation :

1. **Chargement des données nettoyées** depuis le fichier CSV (books_clean.csv).
2. **Distribution des livres par rating** → Nombre de livres pour chaque note attribuée.
3. **Répartition des livres par catégorie de prix** (Low, Medium, High).
4. **Histogramme des prix** → Affichage des prix avec une **courbe KDE** (densité estimée).
5. **Boxplot des prix par rating** → Identification des éventuelles anomalies de prix par niveau de notation.

6. Analyse des catégories :
 - Top 10 des catégories les plus fréquentes
 - Diagramme circulaire pour visualiser la répartition
 - Distribution complète des catégories (graphiques en anglais et en français)
 7. Violin plot des prix par rating → Analyse détaillée de la répartition des prix.
 8. Sauvegarde des graphiques dans le dossier images/ au format PNG (résolution 300 dpi).
-

Bibliothèques utilisées

Le script repose sur **deux bibliothèques majeures de visualisation en Python** :

- **matplotlib** → Gère l'affichage et la personnalisation des graphiques.
- **seaborn** → Simplifie la création de visualisations avancées avec des couleurs et styles améliorés.

Ces bibliothèques permettent de **rendre les graphiques plus lisibles et compréhensibles** tout en mettant en évidence les principales tendances des données.

L'utilisation de **seaborn** permet de générer des visualisations professionnelles avec un **style cohérent et des palettes de couleurs adaptées**.

Grâce aux **histogrammes, boxplots et pie charts**, il est possible **d'explorer rapidement la structure des prix, des notations et des catégories**.

Enfin, **tous les graphiques sont sauvegardés automatiquement** pour une exploitation ultérieure.

📌 Le code source intégral de `data_visualization.py` est disponible ci-dessous.

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Script de visualisation pour Books to Scrape utilisant Seaborn.
5  Ce script charge les données nettoyées depuis le fichier CSV (situé dans le dossier 'output')
6  et génère plusieurs graphiques permettant d'analyser la répartition des livres :
7      - Distribution des livres par rating
8      - Distribution des livres par catégorie de prix
9      - Histogramme des prix avec courbe KDE
10     - Boxplot des prix par rating
11     - Top 10 des catégories de livres (anglais et français)
12     - Diagramme circulaire des 10 principales catégories (anglais et français)
13     - Distribution de toutes les catégories (barres horizontales) (anglais et français)
14     - Violin plot des prix par rating
15     - Boxplot des prix pour le top 10 des catégories (anglais et français)
16 Les graphiques sont affichés et sauvegardés sous forme d'images PNG dans le dossier 'images'
17 avec une résolution de 300 dpi.
18 """
19
20 import os
21 import pandas as pd
22 import matplotlib.pyplot as plt
23 import seaborn as sns
24
25 # Dossier de sortie pour les images
26 image_dir = "images"
27 if not os.path.exists(image_dir):
28     os.makedirs(image_dir)
29
30 def load_data(filename=None):
31     """Charge les données nettoyées depuis le fichier CSV situé dans le dossier 'output'. """
32     if filename is None:
33         filename = os.path.join("output", "books_clean.csv")
34     try:
35         df = pd.read_csv(filename, encoding='utf-8')
36         print("Données chargées pour visualisation depuis", filename)
37         # Création de la colonne rating_int à partir de la colonne rating (convertie en int)
38         if 'rating' in df.columns:
39             df['rating_int'] = df['rating'].astype(int)
40         return df
41     except Exception as e:
42         print("Erreur lors du chargement des données :", e)
43         return None
44
45 def plot_rating_distribution(df):
46     """Graphique en barres pour la distribution des livres par rating. """
47     plt.figure(figsize=(8, 6))
48     sns.set_style("whitegrid")
49     ax = sns.countplot(x='rating_int', data=df, hue='rating_int', palette="viridis",
50                         order=sorted(df['rating_int'].unique()), dodge=False)
51     ax.set_title("Nombre de livres par rating", fontsize=14)
52     ax.set_xlabel("Rating", fontsize=12)
53     ax.set_ylabel("Nombre de livres", fontsize=12)
54     legend = ax.get_legend()
55     if legend is not None:
56         legend.remove()
57     plt.tight_layout()
58     plt.savefig(os.path.join(image_dir, "rating_distribution_seaborn.png"), dpi=300)
59     plt.show()
60
61 def plot_price_category_distribution(df):
62     """Graphique en barres pour la distribution des livres par catégorie de prix. """
63     plt.figure(figsize=(8, 6))
64     sns.set_style("whitegrid")
65     ax = sns.countplot(x='price_category', data=df, hue='price_category', palette="Set2",
66                         order=['Low', 'Medium', 'High'], dodge=False)
67     ax.set_title("Nombre de livres par catégorie de prix", fontsize=14)
68     ax.set_xlabel("Catégorie de prix", fontsize=12)
69     ax.set_ylabel("Nombre de livres", fontsize=12)
70     legend = ax.get_legend()
71     if legend is not None:
72         legend.remove()
73     plt.tight_layout()
74     plt.savefig(os.path.join(image_dir, "price_category_distribution_seaborn.png"), dpi=300)
75     plt.show()

```

```

76
77 def plot_price_histogram(df):
78     """Histogramme des prix avec courbe KDE."""
79     plt.figure(figsize=(8, 6))
80     sns.set_style("whitegrid")
81     ax = sns.histplot(df['price'], bins=30, kde=True, color='skyblue', edgecolor='black')
82     ax.set_title("Histogramme des prix des livres", fontsize=14)
83     ax.set_xlabel("Prix", fontsize=12)
84     ax.set_ylabel("Fréquence", fontsize=12)
85     plt.tight_layout()
86     plt.savefig(os.path.join(image_dir, "price_histogram_seaborn.png"), dpi=300)
87     plt.show()
88
89 def plot_boxplot_price_by_rating(df):
90     """Boxplot des prix par rating."""
91     plt.figure(figsize=(8, 6))
92     sns.set_style("whitegrid")
93     ax = sns.boxplot(x='rating_int', y='price', data=df, hue='rating_int', palette="Set3",
94                      order=sorted(df['rating_int'].unique()), dodge=False)
95     ax.set_title("Boxplot des prix par rating", fontsize=14)
96     ax.set_xlabel("Rating", fontsize=12)
97     ax.set_ylabel("Prix", fontsize=12)
98     legend = ax.get_legend()
99     if legend is not None:
100         legend.remove()
101     plt.tight_layout()
102     plt.savefig(os.path.join(image_dir, "boxplot_price_by_rating_seaborn.png"), dpi=300)
103     plt.show()
104
105 # Graphiques utilisant les catégories en anglais
106 def plot_top_categories_distribution_en(df):
107     """Graphique en barres pour le top 10 des catégories (anglais)."""
108     plt.figure(figsize=(10, 6))
109     sns.set_style("whitegrid")
110     cat_counts = df['category'].value_counts().nlargest(10).reset_index()
111     cat_counts.columns = ['category', 'count']
112     ax = sns.barplot(x='category', y='count', data=cat_counts, hue='category',
113                      palette="coolwarm", dodge=False)
114     ax.set_title("Top 10 des catégories de livres (Anglais)", fontsize=14)
115     ax.set_xlabel("Catégorie", fontsize=12)
116     ax.set_ylabel("Nombre de livres", fontsize=12)
117     legend = ax.get_legend()
118     if legend is not None:
119         legend.remove()
120     plt.xticks(rotation=45, ha='right')
121     plt.tight_layout()
122     plt.savefig(os.path.join(image_dir, "top_categories_distribution_en.png"), dpi=300)
123     plt.show()
124
125 def plot_category_pie_chart_en(df):
126     """Diagramme circulaire pour le top 10 des catégories (anglais)."""
127     plt.figure(figsize=(8, 8))
128     cat_counts = df['category'].value_counts().nlargest(10)
129     plt.pie(cat_counts.values, labels=cat_counts.index, autopct='%1.1f%%', startangle=140,
130             colors=sns.color_palette("pastel"))
131     plt.title("Répartition des 10 principales catégories (Anglais)", fontsize=14)
132     plt.tight_layout()
133     plt.savefig(os.path.join(image_dir, "category_pie_chart_en.png"), dpi=300)
134     plt.show()

```

```

135
136     def plot_all_categories_distribution_en(df):
137         """Graphique en barres horizontales pour toutes les catégories (anglais)."""
138         plt.figure(figsize=(10, 12))
139         sns.set_style("whitegrid")
140         cat_counts = df['category'].value_counts().sort_values(ascending=True).reset_index()
141         cat_counts.columns = ['category', 'count']
142         ax = sns.barplot(x='count', y='category', data=cat_counts, hue='category',
143                           palette="viridis", dodge=False)
144         ax.set_title("Distribution de toutes les catégories de livres (Anglais)", fontsize=14)
145         ax.set_xlabel("Nombre de livres", fontsize=12)
146         ax.set_ylabel("Catégorie", fontsize=12)
147         legend = ax.get_legend()
148         if legend is not None:
149             legend.remove()
150         plt.tight_layout()
151         plt.savefig(os.path.join(image_dir, "all_categories_distribution_en.png"), dpi=300)
152         plt.show()
153
154     def plot_boxplot_price_by_category_en(df):
155         """Boxplot des prix pour le top 10 des catégories (anglais)."""
156         plt.figure(figsize=(12, 6))
157         sns.set_style("whitegrid")
158         top10 = df['category'].value_counts().nlargest(10).index
159         subset = df[df['category'].isin(top10)]
160         ax = sns.boxplot(x='category', y='price', data=subset, hue='category', palette="Set1", dodge=False)
161         ax.set_title("Boxplot des prix pour le top 10 des catégories (Anglais)", fontsize=14)
162         ax.set_xlabel("Catégorie", fontsize=12)
163         ax.set_ylabel("Prix", fontsize=12)
164         legend = ax.get_legend()
165         if legend is not None:
166             legend.remove()
167         plt.xticks(rotation=45, ha='right')
168         plt.tight_layout()
169         plt.savefig(os.path.join(image_dir, "boxplot_price_by_category_en.png"), dpi=300)
170         plt.show()
171
172     # Graphiques utilisant les catégories en français
173     def plot_top_categories_distribution_fr(df):
174         """Graphique en barres pour le top 10 des catégories (français)."""
175         plt.figure(figsize=(10, 6))
176         sns.set_style("whitegrid")
177         cat_counts = df['category_fr'].value_counts().nlargest(10).reset_index()
178         cat_counts.columns = ['category_fr', 'count']
179         ax = sns.barplot(x='category_fr', y='count', data=cat_counts, hue='category_fr',
180                           palette="coolwarm", dodge=False)
181         ax.set_title("Top 10 des catégories de livres (Français)", fontsize=14)
182         ax.set_xlabel("Catégorie (FR)", fontsize=12)
183         ax.set_ylabel("Nombre de livres", fontsize=12)
184         legend = ax.get_legend()
185         if legend is not None:
186             legend.remove()
187         plt.xticks(rotation=45, ha='right')
188         plt.tight_layout()
189         plt.savefig(os.path.join(image_dir, "top_categories_distribution_fr.png"), dpi=300)
190         plt.show()
191
192     def plot_category_pie_chart_fr(df):
193         """Diagramme circulaire pour le top 10 des catégories (français)."""
194         plt.figure(figsize=(8, 8))
195         cat_counts = df['category_fr'].value_counts().nlargest(10)
196         plt.pie(cat_counts.values, labels=cat_counts.index, autopct='%1.1f%%', startangle=140,
197                 colors=sns.color_palette("pastel"))
198         plt.title("Répartition des 10 principales catégories (Français)", fontsize=14)
199         plt.tight_layout()
200         plt.savefig(os.path.join(image_dir, "category_pie_chart_fr.png"), dpi=300)
201         plt.show()

```

```

202
203 def plot_all_categories_distribution_fr(df):
204     """Graphique en barres horizontales pour toutes les catégories (français)."""
205     plt.figure(figsize=(10, 12))
206     sns.set_style("whitegrid")
207     cat_counts = df['category_fr'].value_counts().sort_values(ascending=True).reset_index()
208     cat_counts.columns = ['category_fr', 'count']
209     ax = sns.barplot(x='count', y='category_fr', data=cat_counts, hue='category_fr',
210                       palette="viridis", dodge=False)
211     ax.set_title("Distribution de toutes les catégories de livres (Français)", fontsize=14)
212     ax.set_xlabel("Nombre de livres", fontsize=12)
213     ax.set_ylabel("Catégorie (FR)", fontsize=12)
214     legend = ax.get_legend()
215     if legend is not None:
216         legend.remove()
217     plt.tight_layout()
218     plt.savefig(os.path.join(image_dir, "all_categories_distribution_fr.png"), dpi=300)
219     plt.show()
220
221 def plot_boxplot_price_by_category_fr(df):
222     """Boxplot des prix pour le top 10 des catégories (français)."""
223     plt.figure(figsize=(12, 6))
224     sns.set_style("whitegrid")
225     top10 = df['category_fr'].value_counts().nlargest(10).index
226     subset = df[df['category_fr'].isin(top10)]
227     ax = sns.boxplot(x='category_fr', y='price', data=subset, hue='category_fr', palette="Set1", dodge=False)
228     ax.set_title("Boxplot des prix pour le top 10 des catégories (Français)", fontsize=14)
229     ax.set_xlabel("Catégorie (FR)", fontsize=12)
230     ax.set_ylabel("Prix", fontsize=12)
231     legend = ax.get_legend()
232     if legend is not None:
233         legend.remove()
234     plt.xticks(rotation=45, ha='right')
235     plt.tight_layout()
236     plt.savefig(os.path.join(image_dir, "boxplot_price_by_category_fr.png"), dpi=300)
237     plt.show()
238
239 def plot_violin_price_by_rating(df):
240     """Violin plot des prix par rating."""
241     plt.figure(figsize=(8, 6))
242     sns.set_style("whitegrid")
243     ax = sns.violinplot(x='rating_int', y='price', data=df, hue='rating_int', palette="Set2", dodge=False)
244     ax.set_title("Violin plot des prix par rating", fontsize=14)
245     ax.set_xlabel("Rating", fontsize=12)
246     ax.set_ylabel("Prix", fontsize=12)
247     legend = ax.get_legend()
248     if legend is not None:
249         legend.remove()
250     plt.tight_layout()
251     plt.savefig(os.path.join(image_dir, "violin_price_by_rating.png"), dpi=300)
252     plt.show()
253
254 def main():
255     df = load_data() # Charge depuis 'output/books_clean.csv'
256     if df is None:
257         return
258     plot_rating_distribution(df)
259     plot_price_category_distribution(df)
260     plot_price_histogram(df)
261     plot_boxplot_price_by_rating(df)
262     # Graphiques en anglais
263     plot_top_categories_distribution_en(df)
264     plot_category_pie_chart_en(df)
265     plot_all_categories_distribution_en(df)
266     plot_boxplot_price_by_category_en(df)
267     # Graphiques en français
268     plot_top_categories_distribution_fr(df)
269     plot_category_pie_chart_fr(df)
270     plot_all_categories_distribution_fr(df)
271     plot_boxplot_price_by_category_fr(df)
272     # Autre graphique commun
273     plot_violin_price_by_rating(df)
274
275 if __name__ == '__main__':
276     main()

```

8.1.4 Script Principal et Menu Interactif (main.py)

Le fichier `main.py` est le point d'entrée du projet. Il propose une **interface interactive en ligne de commande** permettant **d'exécuter facilement chaque étape du pipeline ETL**.

L'utilisateur peut ainsi choisir d'effectuer :

1. **Le scraping et l'insertion des données en base**
2. **Le nettoyage et l'analyse des données**
3. **La génération des visualisations**
4. **L'exécution complète du pipeline ETL** (toutes les étapes enchaînées)
5. **La sortie du programme**

Le menu est affiché sous une **forme intuitive et dynamique**, offrant une meilleure expérience utilisateur grâce à la bibliothèque `rich`, qui permet **d'afficher des panneaux colorés** pour rendre le terminal plus attrayant.

Fonctionnalités et structure du script :

- Utilisation de rich pour améliorer l'affichage des messages** (panneaux colorés pour indiquer l'état des processus).
- Exécution conditionnelle des modules** : chaque option du menu exécute **le script correspondant** (`scraping.py`, `data_cleaning_analysis.py`, `data_visualization.py`).
- Gestion du pipeline ETL complet** → Exécution séquentielle de toutes les étapes avec de **courtes pauses** (`time.sleep()`) pour améliorer la lisibilité.
- Boucle interactive** → L'utilisateur peut choisir son action en saisissant un numéro (1 à 5) pour interagir avec le programme.

Bibliothèques utilisées

Le script utilise principalement :

- `time` → Gestion des pauses entre les étapes pour améliorer la lisibilité des exécutions successives.
 - `rich` → Amélioration visuelle de l'affichage des menus et des messages en terminal.
 - **Les fichiers du projet** → Importation des modules `scraping.py`, `data_cleaning_analysis.py` et `data_visualization.py` pour permettre leur exécution directe depuis le menu.
-

L'utilisation de `rich` rend l'expérience **beaucoup plus ergonomique et agréable**, en ajoutant des panneaux colorés pour **indiquer clairement les actions en cours et les statuts d'exécution**. Grâce à ce menu interactif, **toutes les fonctionnalités du projet peuvent être exécutées simplement et efficacement**, sans avoir à lancer manuellement chaque script.

 **Le code source intégral de main.py est disponible ci-dessous.**

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Main interactive script for BooksToScrape_ETL.
5
6  This script provides an interactive menu to run:
7  1. Scraping and Database Insertion
8  2. Data Cleaning and Analysis
9  3. Data Visualization
10 4. Full ETL Pipeline
11 5. Exit
12
13 It uses the rich library to display colorful messages.
14 """
15
16 import time
17 from rich.console import Console
18 from rich.panel import Panel
19
20 # Import project modules (ensure these are in the same directory)
21 import scraping
22 import data_cleaning_analysis
23 import data_visualization
24
25 console = Console()
26
27
28 def run_scraping():
29     console.print(Panel("Starting Scraping Process...", style="bold green"))
30     scraping.main()
31     console.print(Panel("Scraping completed successfully.", style="bold green"))
32
33
34 def run_cleaning_analysis():
35     console.print(Panel("Starting Data Cleaning and Analysis...", style="bold blue"))
36     data_cleaning_analysis.main()
37     console.print(Panel("Data Cleaning and Analysis completed successfully.", style="bold blue"))
38
39
40 def run_visualization():
41     console.print(Panel("Starting Data Visualization...", style="bold magenta"))
42     data_visualization.main()
43     console.print(Panel("Data Visualization completed successfully.", style="bold magenta"))
44
45
46 def run_full_pipeline():
47     console.print(Panel("Running Full ETL Pipeline...", style="bold yellow"))
48     run_scraping()
49     time.sleep(1)
50     run_cleaning_analysis()
51     time.sleep(1)
52     run_visualization()
53     time.sleep(1)
54     console.print(Panel("Full ETL Pipeline executed successfully!", style="bold green"))

```

```

55
56
57 def interactive_menu():
58     while True:
59         console.print(Panel("BooksToScrape_ETL Interactive Menu", style="bold underline cyan"))
60         console.print("[1] Run Scraping and Database Insertion")
61         console.print("[2] Run Data Cleaning and Analysis")
62         console.print("[3] Run Data Visualization")
63         console.print("[4] Run Full ETL Pipeline")
64         console.print("[5] Exit")
65
66         choice = input("Enter your choice (1-5): ").strip()
67         if choice == "1":
68             run_scraping()
69         elif choice == "2":
70             run_cleaning_analysis()
71         elif choice == "3":
72             run_visualization()
73         elif choice == "4":
74             run_full_pipeline()
75         elif choice == "5":
76             console.print(Panel("Exiting. Goodbye!", style="bold red"))
77             break
78         else:
79             console.print("[red]Invalid choice. Please try again.[/red]")
80             console.print("\n")
81             time.sleep(1)
82
83
84 def main():
85     interactive_menu()
86
87
88 if __name__ == '__main__':
89     main()
90

```

8.1.5 Script SQL de la Base de Données (database.sql)

Le fichier **database.sql** contient les instructions **nécessaires à la gestion de la base de données** utilisée pour stocker les informations extraites du site **Books To Scrape**. Ce script SQL permet de **créer la base de données, définir la structure de la table books, insérer les données et vérifier la cohérence du stockage**.

Fonctionnalités et structure du script :

- Création de la base de données** `books_scrape`, avec une gestion **optimisée des caractères Unicode** (`utf8mb4`).
- Définition de la table books** avec :
 - `id` → Identifiant unique auto-incrémenté (`PRIMARY KEY`).
 - `title` → Titre du livre (`VARCHAR(255) NOT NULL`).
 - `price` → Prix du livre, stocké en **décimal** avec deux chiffres après la virgule (`DECIMAL(6,2)`).
 - `rating` → Note du livre (`INT NOT NULL`).
 - `product_link` → Lien vers la fiche produit (`VARCHAR(255) NOT NULL`).
 - `category` → Catégorie du livre (`VARCHAR(255)`, valeur optionnelle).
- Interrogation et vérification des données** :
 - `SHOW TABLES;` pour afficher les tables de la base de données.
 - `DESCRIBE books;` pour **visualiser la structure de la table books**.

- `SELECT * FROM books;` pour afficher l'ensemble des données enregistrées.
-

Optimisation et choix techniques :

- **Encodage en utf8mb4_unicode_ci** → Permet d'assurer la compatibilité avec tous les caractères spéciaux et accents présents dans les titres des livres.
 - **Stockage du prix en DECIMAL(6,2)** → Évite les erreurs d'arrondi pouvant survenir avec le type FLOAT.
 - **Indexation automatique sur id** → Améliore la performance des requêtes sur la table.
-

Pourquoi une base MySQL ?

L'utilisation d'une **base de données relationnelle** comme MySQL permet de **structurer et d'organiser efficacement les données**, facilitant ainsi :

- ◆ **Les recherches avancées** (`SELECT * FROM books WHERE price > 20;`).
- ◆ **L'analyse statistique** (`AVG(price)` pour calculer le prix moyen).
- ◆ **La gestion et la persistance des données**, contrairement aux fichiers CSV ou JSON.

Une fois les données **scrapées et nettoyées**, elles sont **insérées automatiquement** dans cette base, garantissant une **exploitation simplifiée** pour des analyses ultérieures.

✖ Le code source intégral de `database.sql` est disponible ci-dessous.

```
● ● ●  
1  -- Active: 1737537456443@127.0.0.1@3306@books_scrape  
2  -- Création de la base de données avec gestion des caractères Unicode  
3  CREATE DATABASE IF NOT EXISTS books_scrape  
4      DEFAULT CHARACTER SET utf8mb4  
5      COLLATE utf8mb4_unicode_ci;  
6  
7  -- Sélection de la base de données  
8  USE books_scrape;  
9  
10 -- Création de la table "books"  
11 CREATE TABLE IF NOT EXISTS books (  
12     id INT AUTO_INCREMENT PRIMARY KEY,  
13     title VARCHAR(255) NOT NULL,  
14     price DECIMAL(6,2) NOT NULL,  
15     rating INT NOT NULL,  
16     product_link VARCHAR(255) NOT NULL,  
17     category VARCHAR(255) DEFAULT NULL  
18 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
19  
20 -- Pour afficher les tables et leur contenu :  
21 SHOW TABLES;  
22 DESCRIBE books;  
23 SELECT * FROM books;
```

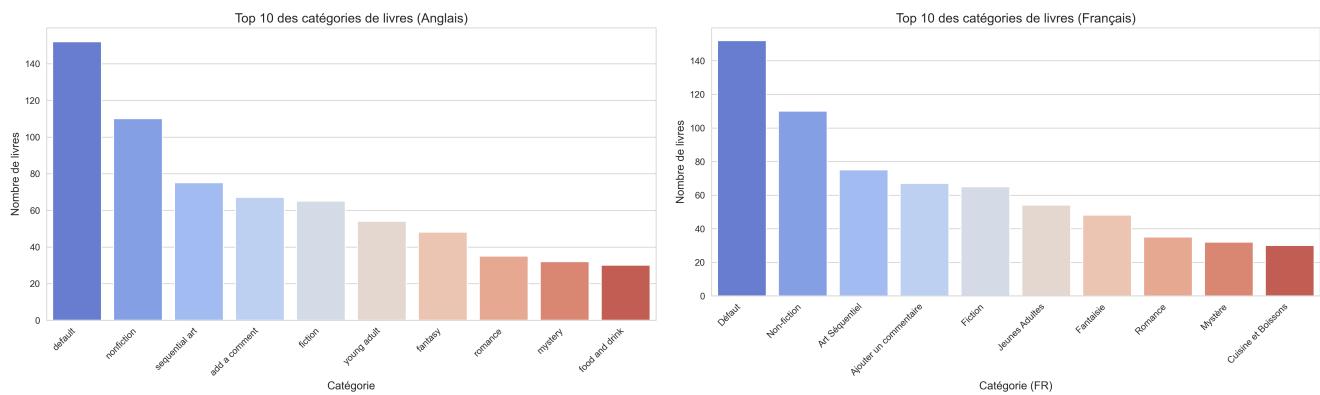
8.2. Visualisations Supplémentaires

Cette section regroupe **toutes les visualisations générées** par le script `data_visualization.py`. Ces graphiques permettent d'explorer **les tendances des prix, des notations et des catégories** des livres extraits et nettoyés.

Chaque graphique apporte une **analyse complémentaire**, facilitant la compréhension des **données collectées** et des **corrélations** éventuelles.

8.2.1 Distribution des livres par catégorie (Anglais et Français)

Images : `top_categories_distribution_en.png` et `top_categories_distribution_fr.png`



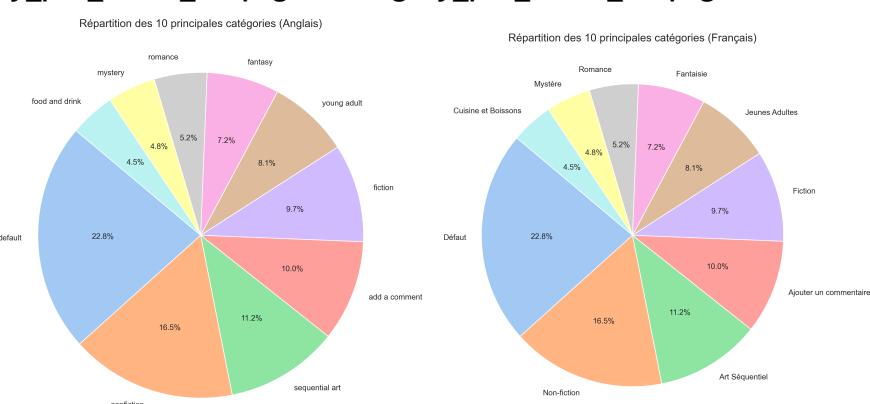
Ces histogrammes affichent la répartition des livres par **catégorie**, à la fois en **anglais** et après traduction en **français**. Ils permettent de voir **quelles catégories sont les plus représentées** sur la plateforme Books To Scrape.

Ce que montrent ces graphiques :

- Les **10 catégories les plus fréquentes** dans l'ensemble des données collectées.
 - Une comparaison entre les **catégories originales** et leur **équivalent traduit en français**.
 - Une vision claire des **genres dominants**, facilitant l'analyse des tendances du marché du livre sur ce site.
-

8.2.2 Répartition des catégories sous forme de diagramme circulaire

Images : `category_pie_chart_en.png` et `category_pie_chart_fr.png`



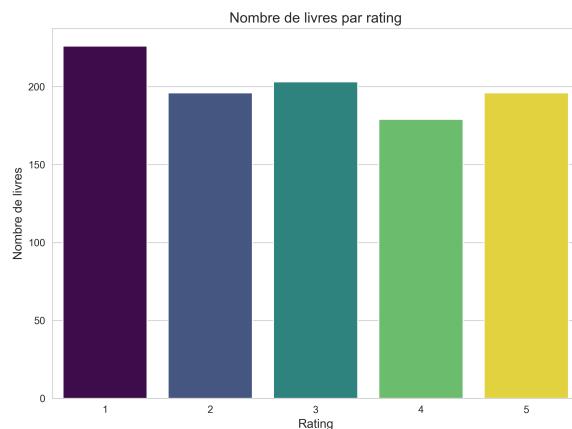
Ces **diagrammes circulaires** (camemberts) montrent la **répartition des livres par catégorie**. Ils complètent la visualisation précédente en mettant en évidence la **proportion relative** de chaque catégorie.

Ce que montrent ces graphiques :

- Une **vue synthétique** de l'importance de chaque catégorie.
 - La présence éventuelle de **catégories sous-représentées**.
 - Une comparaison entre la version **anglaise et française** des données après nettoyage et traduction.
-

8.2.3 Répartition des livres selon leur notation (rating)

Image : rating_distribution_seaborn.png



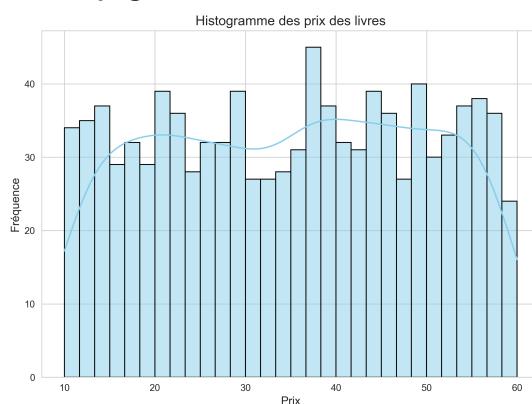
Ce graphique en **barres** montre la **distribution des livres en fonction de leur notation**. Il est utile pour voir **si les notes sont équilibrées** ou si certains niveaux sont surreprésentés.

Ce que montre ce graphique :

- Une vue globale sur la répartition des notes des livres.
 - Un éventuel biais du site Books To Scrape, avec peut-être une **surreprésentation des livres bien notés**.
 - La **fréquence des notes basses** par rapport aux notes élevées.
-

8.2.4 Distribution des prix des livres

Image : price_histogram_seaborn.png



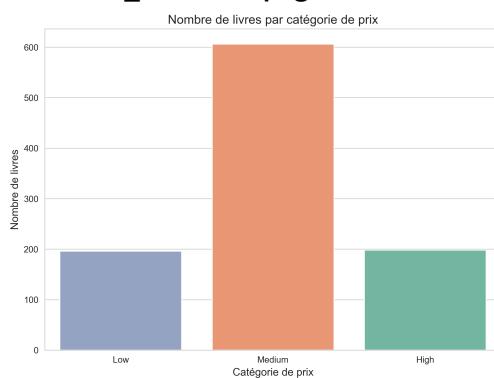
Cet **histogramme des prix** permet d'analyser comment les prix sont répartis dans l'ensemble des données extraites. Il inclut également une **courbe KDE** qui met en évidence les tendances de densité des prix.

Ce que montre ce graphique :

- L'**échelle des prix** majoritairement observés.
 - La présence éventuelle de **valeurs extrêmes**, c'est-à-dire des livres très chers ou très bon marché.
 - Une vision claire des prix les plus fréquents, avec une **distribution en cloche** qui montre une tendance centrale.
-

8.2.5 Répartition des livres par catégorie de prix

 **Image : price_category_distribution_seaborn.png**



Ce graphique regroupe les livres selon **trois catégories de prix** :

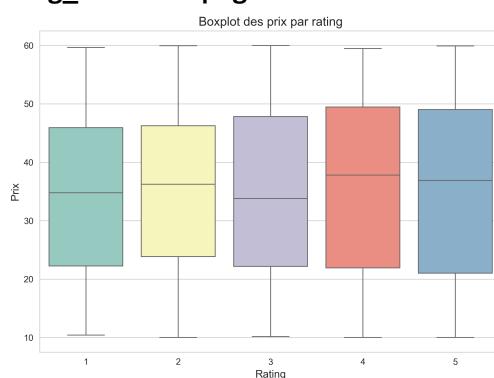
- **Low** (prix bas)
- **Medium** (prix moyen)
- **High** (prix élevé)

Ce que montre ce graphique :

- Quelle proportion des livres appartient à **chaque catégorie de prix**.
 - Si les livres sont plutôt accessibles ou s'il existe une forte disparité de prix.
 - Un aperçu de la **structure tarifaire du site Books To Scrape**.
-

8.2.6 Boxplot des prix par rating

 **Image : boxplot_price_by_rating_seaborn.png**



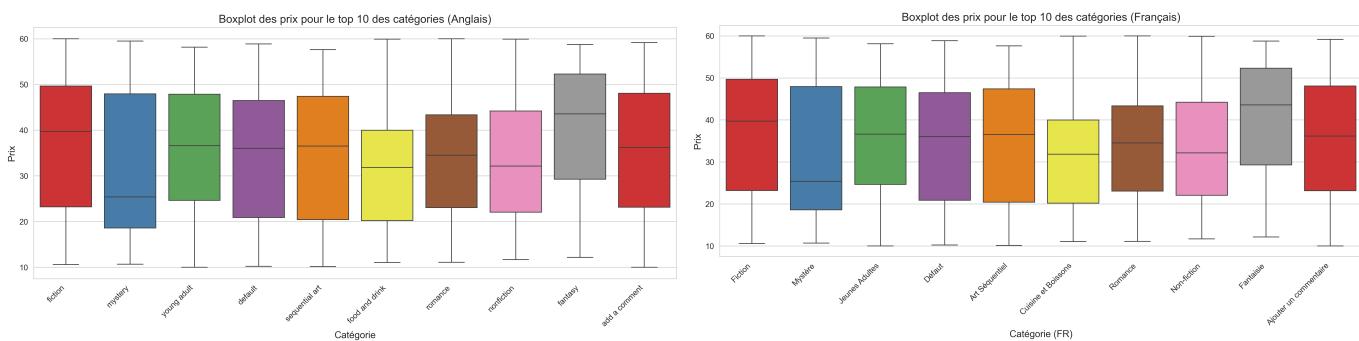
Le **boxplot** permet d'analyser la **répartition des prix** en fonction des **notations des livres**.

Ce que montre ce graphique :

- Si les livres mieux notés ont tendance à être plus chers.
- La **médiane des prix** pour chaque niveau de notation.
- La présence éventuelle de **valeurs aberrantes**, qui sont des livres très chers par rapport aux autres.

8.2.7 Boxplot des prix par catégorie (Anglais et Français)

✖ **Images : boxplot_price_by_category_en.png et boxplot_price_by_category_fr.png**



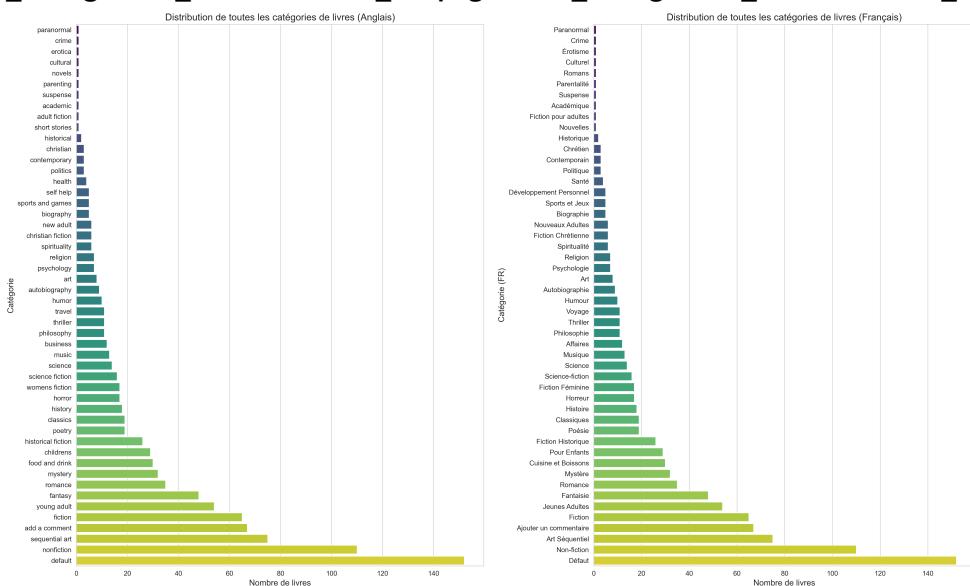
Ces deux **boxplots** montrent la **répartition des prix** pour les **10 catégories les plus fréquentes**.

Ce que montrent ces graphiques :

- Les **catégories où les prix sont les plus élevés** et celles où les livres sont plus accessibles.
- La **médiane des prix** pour chaque catégorie.
- Une comparaison entre la **version anglaise et la version traduite** après nettoyage des données.

8.2.8 Répartition de toutes les catégories (Anglais et Français)

✖ **Images : all_categories_distribution_en.png et all_categories_distribution_fr.png**



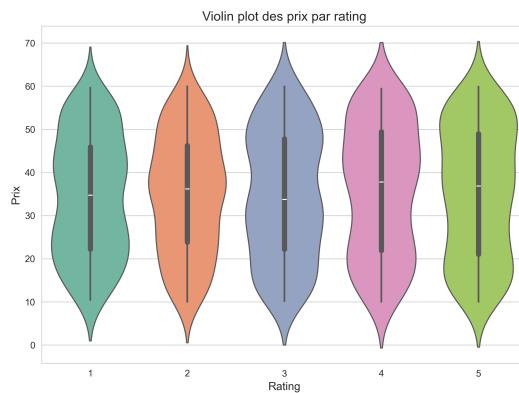
Ces graphiques montrent la répartition **de toutes les catégories présentes dans les données**, sous forme de **barres horizontales**.

Ce que montrent ces graphiques :

- Une vue d'ensemble de **toutes les catégories extraites**.
 - Une **comparaison entre la version originale et la version traduite en français**.
 - Un aperçu des catégories **sous-représentées** ou **dominantes** dans l'ensemble des données.
-

8.2.9 Violin plot des prix par rating

 **Image : violin_price_by_rating.png**



Le **violin plot** est une version améliorée du **boxplot**, qui montre non seulement la distribution des prix, mais aussi la **densité des données**.

Ce que montre ce graphique :

- Comment les prix sont répartis selon chaque **niveau de notation**.
 - La présence de **valeurs aberrantes**, qui sont les livres les plus chers.
 - Une visualisation détaillée des **gammes de prix pour chaque rating**.
-

8.3. Explications Techniques Supplémentaires

Cette section apporte **des précisions techniques** sur certains aspects du projet.

8.3.1 Optimisation du Scraping avec ThreadPoolExecutor

Pour accélérer l'extraction des données, nous avons utilisé **ThreadPoolExecutor**, une approche qui permet d'exécuter plusieurs requêtes en parallèle plutôt que de les traiter séquentiellement. Cette technique améliore **significativement la vitesse de scraping** en exploitant plusieurs threads, réduisant ainsi le temps d'exécution total.

Contrairement à une exécution classique, où chaque page est téléchargée l'une après l'autre, **ThreadPoolExecutor permet de récupérer plusieurs pages simultanément**. Cette approche est particulièrement efficace pour traiter **de grands volumes de données** et optimiser l'utilisation des ressources réseau.

📌 Extrait de code montrant l'utilisation de ThreadPoolExecutor dans scraping.py

```
 1  from concurrent.futures import ThreadPoolExecutor
 2
 3  def fetch_all_pages_concurrent(base_url):
 4      """
 5          Parcourt toutes les pages du site en parallèle en suivant la pagination.
 6          Retourne la liste complète des livres extraits.
 7      """
 8      first_html, first_effective_url = fetch_page(base_url)
 9      if not first_html:
10          return []
11
12      soup = BeautifulSoup(first_html, 'html.parser')
13      current_page_text = soup.find("li", class_="current")
14
15      num_pages = 50 # Valeur par défaut
16      if current_page_text:
17          try:
18              num_pages = int(current_page_text.get_text(strip=True).split()[-1])
19          except ValueError:
20              print("Impossible d'extraire le nombre de pages, utilisation de 50 par défaut.")
21
22      # Générer la liste des URLs des pages à scraper
23      urls = [base_url] + [f"http://books.toscrape.com/catalogue/page-{i}.html" for i in range(2, num_pages + 1)]
24
25      books = []
26      with ThreadPoolExecutor(max_workers=20) as executor:
27          results = list(executor.map(fetch_page, urls))
28
29          # Utilisation de tqdm pour afficher la progression dans le traitement des pages
30          for html, effective_url in tqdm(results, total=len(results), desc="Traitement des pages listing"):
31              if html:
32                  books.extend(parse_books(html, effective_url))
33
34  return books
```

Explication de l'optimisation avec ThreadPoolExecutor

Dans ce projet, **le scraping classique est séquentiel** : chaque page est extraite l'une après l'autre, ce qui peut être très lent lorsque **plusieurs dizaines de pages** doivent être traitées. Pour améliorer **les performances**, on utilise **ThreadPoolExecutor**, qui permet de **traiter plusieurs pages en parallèle**.

✓ Avantages de cette approche :

- **Réduction du temps de scraping** grâce à l'exécution concurrente.
- **Utilisation optimale des ressources réseau**, puisque plusieurs requêtes sont envoyées simultanément.
- **Amélioration de la fluidité du processus** en exploitant un **nombre configurable de threads** (`max_workers=20` dans notre cas).

💡 Pourquoi 20 threads ?

Nous avons fixé `max_workers=20` après plusieurs tests pour trouver un bon compromis entre **vitesse et stabilité**. Une valeur trop élevée risquerait de **surcharger le serveur** ou de provoquer des **timeouts**.

📌 Résumé

En appliquant cette approche, le scraping complet du site **Books to Scrape** est **2 à 3 fois plus rapide** qu'une méthode séquentielle, tout en garantissant une récupération efficace des données.

8.3.2 Conversion et Nettoyage des Données avec Pandas

Afin d'assurer la qualité et l'exploitabilité des données extraites, plusieurs étapes de nettoyage et de transformation ont été réalisées avec **Pandas**. L'objectif principal était d'homogénéiser les données, d'éliminer les incohérences et de préparer un dataset propre pour les analyses et visualisations.

✖ Principales opérations effectuées :

- **Suppression des caractères spéciaux** : Le prix des livres est initialement stocké sous forme de texte contenant le symbole "£". Pour permettre son exploitation numérique, il est nettoyé et converti en type **float**.
- **Gestion des valeurs manquantes** : Certaines valeurs sont absentes, notamment dans la colonne des **catégories**. Nous avons donc appliqué **fillna()** pour éviter toute incohérence.
- **Conversion des types** : Les données sont converties dans des formats adaptés, avec **rating** en **int** et **price** en **float**, garantissant ainsi une meilleure manipulation des données.

✖ Extrait de code montrant ces opérations dans `data_cleaning_analysis.py`

```
 1  def clean_data(df):
 2      """Nettoyage et préparation des données."""
 3      print("Premières lignes des données brutes:")
 4      print(df.head())
 5
 6      # Affichage des valeurs manquantes initiales
 7      print("\nValeurs manquantes par colonne AVANT nettoyage:")
 8      print(df.isnull().sum())
 9
10     # Gestion des valeurs manquantes
11     df['title'] = df['title'].fillna("inconnu")
12     df['price'] = df['price'].fillna(0)
13     df['rating'] = df['rating'].fillna(0)
14     df['product_link'] = df['product_link'].fillna("inconnu")
15     df['category'] = df['category'].fillna("inconnu")
16
17     # Conversion des types de données
18     # Suppression du symbole '£' et conversion du prix en float
19     if df['price'].dtype == object:
20         df['price'] = df['price'].replace({'£': ''}, regex=True)
21         df['price'] = pd.to_numeric(df['price'], errors='coerce').fillna(0)
22
23     # Conversion du rating en entier
24     df['rating'] = pd.to_numeric(df['rating'], errors='coerce').fillna(0).astype(int)
25
26     return df
```

Explication des étapes du nettoyage

Suppression du symbole monétaire

Le site Books To Scrape stocke les prix avec le symbole "£". Pour effectuer des calculs, **regex est utilisé pour nettoyer la colonne "price"** en supprimant ce symbole et en convertissant les valeurs en **float**.

Traitement des valeurs manquantes

- `fillna()` est appliqué sur **title**, **price**, **rating**, **product_link** et **category** afin d'éviter tout problème lors des traitements ultérieurs.
- Par défaut, un livre sans titre est nommé "**inconnu**", un prix vide est **0**, et une catégorie non renseignée est également remplacée par "**inconnu**".

Conversion des types

- **Prix en float** : nécessaire pour les analyses statistiques et les visualisations.
- **Notation en int** : initialement récupérée sous forme de texte, elle est convertie en **entier** pour des comparaisons plus simples.

Résumé

Grâce à ces opérations de nettoyage, les données deviennent **plus cohérentes et directement exploitables** pour l'analyse et la visualisation.

8.3.3 Explication des Visualisations

Les **visualisations graphiques** ont été choisies pour extraire des tendances et rendre l'analyse des données plus accessible. Grâce aux bibliothèques **Matplotlib** et **Seaborn**, plusieurs types de représentations ont été générés afin d'explorer différentes facettes du dataset.

Principaux types de visualisation utilisés :

Histogrammes et courbes KDE (Kernel Density Estimation)

L'histogramme des prix montre comment les livres sont distribués en fonction de leur coût. Une **courbe KDE** est ajoutée pour représenter la **densité de distribution** de manière plus fluide et intuitive.

Boxplots et Violin Plots

Ces graphiques permettent d'analyser la **dispersion des prix en fonction des notations** attribuées aux livres.

- Le **Boxplot** met en évidence les **quartiles, médianes et valeurs extrêmes (outliers)**.
- Le **Violin Plot** ajoute une distribution **lissée** pour mieux visualiser la concentration des valeurs.

Diagrammes circulaires et bar charts

Ces représentations sont utilisées pour visualiser la **répartition des catégories de livres en anglais et français**.

- Les **bar charts** classent les catégories par fréquence.
- Les **diagrammes circulaires (pie charts)** permettent une vue plus compacte des **top 10 catégories les plus fréquentes**.

Comparaison des prix par catégorie et rating

- Les **Boxplots des prix par catégorie** permettent de comparer **l'amplitude et la médiane** des prix entre les différentes catégories.
- Les **Boxplots des prix par notation** offrent une perspective sur la variation des prix en fonction des **évaluations attribuées aux livres**.

📌 Extrait de code générant un des graphiques depuis data_visualization.py

```
● ● ●  
1 def plot_price_histogram(df):  
2     """Histogramme des prix avec courbe KDE."""  
3     plt.figure(figsize=(8, 6))  
4     sns.set_style("whitegrid")  
5     ax = sns.histplot(df['price'], bins=30, kde=True, color='skyblue', edgecolor='black')  
6     ax.set_title("Histogramme des prix des livres", fontsize=14)  
7     ax.set_xlabel("Prix", fontsize=12)  
8     ax.set_ylabel("Fréquence", fontsize=12)  
9     plt.tight_layout()  
10    plt.savefig(os.path.join(image_dir, "price_histogram_seaborn.png"), dpi=300)  
11    plt.show()
```

Pourquoi ces visualisations ?

🔍 Analyse des tendances

- L'**histogramme** met en évidence la **répartition des prix**, permettant d'identifier d'éventuels **pics de concentration**.
- Les **courbes KDE** aident à comprendre **où se situent les prix les plus fréquents**.

📊 Comparaison des catégories et notations

- Les **barres horizontales** facilitent la lecture des **différences entre catégories de livres**.
- Les **boxplots** permettent d'analyser la **dispersion et la médiane des prix** selon différentes variables (rating, catégorie, etc.).

📌 Résumé

Ces visualisations permettent de **résumer visuellement** les données, facilitant l'identification de tendances et **l'interprétation rapide** des résultats.

8.4. Formats d'Exportation des Données

Les données extraites et nettoyées sont exportées sous trois formats pour faciliter leur exploitation et leur analyse :

8.4.1 Export en CSV

	title	price	rating	product_link	category
1	A Light in the Attic	51.77	3	http://books.toscrape.com/catalogue/a-light-in-...	Poetry
2	Tipping the Velvet	53.74	1	http://books.toscrape.com/catalogue/tipping-the...	Historical Fiction
3	Soumission	50.1	1	http://books.toscrape.com/catalogue/soumission_...	Fiction
4	Sharp Objects	47.82	4	http://books.toscrape.com/catalogue/sharp-objec...	Mystery
5	Sapiens: A Brief History of Humankind	54.23	5	http://books.toscrape.com/catalogue/sapiens-a-b...	History
6	The Requiem Red	22.65	1	http://books.toscrape.com/catalogue/the-requiem...	Young Adult
7	The Dirty Little Secrets of Getting Your Dream ...	33.34	4	http://books.toscrape.com/catalogue/the-dirty-l...	Business
8	The Coming Woman: A Novel Based on the Life of ...	17.93	3	http://books.toscrape.com/catalogue/the-coming-w...	Default
9	The Boys in the Boat: Nine Americans and Their ...	22.6	4	http://books.toscrape.com/catalogue/the-boys-in...	Default
10	The Black Maria	52.15	1	http://books.toscrape.com/catalogue/the-black-m...	Poetry

Le format CSV est utilisé pour sa simplicité et son interopérabilité avec la plupart des logiciels d'analyse de données.

- Séparateur utilisé : , (virgule).
- Encodage : UTF-8.
- Exemple de contenu :

```
title,price,rating,product_link,category
A Light in the Attic,51.77,3,http://books.toscrape.com/catalogue/a-light-in-the-
attic_1000/index.html,Poetry
Tipping the Velvet,53.74,1,http://books.toscrape.com/catalogue/tipping-the-
velvet_999/index.html,Historical Fiction
```

8.4.2 Export en JSON

```
1  [
2    {
3      "title": "A Light in the Attic",
4      "price": 51.77,
5      "rating": 3,
6      "product_link": "http://books.toscrape.com/catalogue/a-light-in-the-attic_1000/index.html",
7      "category": "Poetry"
8    },
9    {
10      "title": "Tipping the Velvet",
11      "price": 53.74,
12      "rating": 1,
13      "product_link": "http://books.toscrape.com/catalogue/tipping-the-velvet_999/index.html",
14      "category": "Historical Fiction"
15    },
16    {
17      "title": "Soumission",
18      "price": 50.1,
19      "rating": 1,
20      "product_link": "http://books.toscrape.com/catalogue/soumission_998/index.html",
21      "category": "Fiction"
22    },
23    {
24      "title": "Sharp Objects",
25      "price": 47.82,
26      "rating": 4,
27      "product_link": "http://books.toscrape.com/catalogue/sharp-objects_997/index.html",
28      "category": "Mystery"
29    },
30  ]
```

Le format JSON est privilégié pour son utilisation en développement web et sa compatibilité avec les bases NoSQL.

- Structure de stockage en format **dictionnaire de liste** :

```
{
  "title": "A Light in the Attic",
  "price": 51.77,
  "rating": 3,
  "product_link": "http://books.toscrape.com/catalogue/a-light-in-the-
attic_1000/index.html",
  "category": "Poetry"
},
{
  "title": "Tipping the Velvet",
  "price": 53.74,
  "rating": 1,
  "product_link": "http://books.toscrape.com/catalogue/tipping-the-
velvet_999/index.html",
  "category": "Historical Fiction"
},
```

8.4.3 Export en Excel

C0	C1	C2	C3	C4
title	price	rating	product_link	category
A Light in the Attic	51.77	3	http://books.toscrape.com/catalogue/a-light-in-the...	Poetry
Tipping the Velvet	53.74	1	http://books.toscrape.com/catalogue/tipping-the-ve...	Historical Fiction
Soumission	50.1	1	http://books.toscrape.com/catalogue/soumission_998...	Fiction
Sharp Objects	47.82	4	http://books.toscrape.com/catalogue/sharp-objects...	Mystery
Sapiens: A Brief History of Humankind	54.23	5	http://books.toscrape.com/catalogue/sapiens-a-brie...	History
The Requiem Red	22.65	1	http://books.toscrape.com/catalogue/the-requiem-re...	Young Adult
The Dirty Little Secrets of Getting Your Dream Job	33.34	4	http://books.toscrape.com/catalogue/the-dirty-litt...	Business
The Coming Woman: A Novel Based on the Life of the...	17.93	3	http://books.toscrape.com/catalogue/the-coming-wom...	Default
The Boys in the Boat: Nine Americans and Their Epi...	22.6	4	http://books.toscrape.com/catalogue/the-boys-in-th...	Default

Le format Excel est utilisé pour les analyses avancées dans un tableau et permet de bénéficier des fonctionnalités de filtrage et de tri interactifs.

- Encodage : UTF-8.
- Feuille unique nommée books_clean.
- Colonne ajoutée : price_category (Low, Medium, High).

8.5. Exécution du Programme et Résultats en Console

Cette section présente des captures d'écran illustrant l'exécution des différentes étapes du programme dans l'environnement **PyCharm**. Ces sorties permettent de visualiser le bon déroulement du scraping, du nettoyage des données, de l'insertion en base de données et de la génération des visualisations.

📌 Les images ci-dessous montrent :

- Le lancement du script principal et l'affichage du menu interactif.
- L'exécution du **scraping**, avec l'extraction et l'enregistrement des livres.
- Le **nettoyage des données**, avec les statistiques avant/après traitement.
- La **génération des visualisations**, confirmant le bon fonctionnement des scripts.

The screenshot shows the PyCharm interface with the project 'BooksToScrape_ETL' open. The terminal window displays the following sequence of events:

- The user runs the script: `Running Full ETL Pipeline...`
- The process begins with `Starting Scraping Process...`
- Logs show the start of scraping: `Début du scraping du site : http://books.toscrape.com/`
- The scraping process is shown in progress with multiple parallel tasks, each indicating 100% completion: `Récupération des catégories: 100%`
- Logs show the end of the scraping phase: `Fin du scraping du site : http://books.toscrape.com/`
- The process moves to data cleaning and analysis: `Starting Data Cleaning and Analysis...`
- Logs show the start of data visualization: `Starting Data Visualization...`
- The final log message indicates the completion of the full ETL pipeline: `Fin du pipeline ETL : 20/20 [00:00<00:00, 49.49it/s]`

```

BT BooksToScrape_ETL  main
Project  scraping.py  data_cleaning_analysis.py  data_visualization.py  main.py
Run  main  :
G  :  :
Traitement des pages listing: 100% | 50/50 [00:20<00:00, 2.41it/s]
88 1000 livres trouvés en 22.06 secondes.
{'title': 'A Light in the Attic', 'price': 51.77, 'rating': 3, 'product_link': 'http://books.toscrape.com/catalogue/a-light-in-the-attic_1000/index.html', 'category': 'Poetry'}
... {'title': 'Tipping the Velvet', 'price': 53.74, 'rating': 1, 'product_link': 'http://books.toscrape.com/catalogue/tipping-the-velvet_999/index.html', 'category': 'Historical Fiction'}
... {'title': 'Soumission', 'price': 50.1, 'rating': 1, 'product_link': 'http://books.toscrape.com/catalogue/soumission_998/index.html', 'category': 'Fiction'}
... {'title': 'Sharp Objects', 'price': 47.82, 'rating': 4, 'product_link': 'http://books.toscrape.com/catalogue/sharp-objects_997/index.html', 'category': 'Mystery'}
... {'title': 'Sapiens: A Brief History of Humankind', 'price': 54.23, 'rating': 5, 'product_link': 'http://books.toscrape.com/catalogue/sapiens-a-brief-history-of-humankind_996/index.html', 'category': 'History'}
Les données ont été sauvegardées dans le fichier output/books.csv
Les données ont été sauvegardées dans le fichier output/books.json
Les données ont été sauvegardées dans le fichier output/books.xlsx
Les données ont été insérées dans la base de données MySQL après viderage de la table.

[ Scraping completed successfully. ]

Starting Data Cleaning and Analysis...
Données chargées avec succès depuis output/books.csv
Premières lignes des données brutes:
title  price  rating  product_link  category
0      A Light in the Attic  51.77    3  http://books.toscrape.com/catalogue/a-light-in...  Poetry
1      Tipping the Velvet  53.74    1  http://books.toscrape.com/catalogue/tipping-th...  Historical Fiction
2      Soumission  50.10    1  http://books.toscrape.com/catalogue/soumission...  Fiction
3      Sharp Objects  47.82    4  http://books.toscrape.com/catalogue/sharp-obje...  Mystery
4  Sapiens: A Brief History of Humankind  54.23    5  http://books.toscrape.com/catalogue/sapiens-a...  History

Valeurs manquantes par colonne AVANT nettoyage:
title      0
price      0
rating     0
BooksToScrape_ETL > data_cleaning_analysis.py
... Codeium 105:21 LF UTF-8 4 spaces Python 3.12 (BooksToScrape_ETL) ⌂

```

```

BT BooksToScrape_ETL  main
Project  scraping.py  data_cleaning_analysis.py  data_visualization.py  main.py
Run  main  :
G  :  :
product_link  0
category  0
dtype: int64

Valeurs manquantes par colonne APRÈS nettoyage:
... title      0
... price      0
... rating     0
... product_link  0
... category    0
... title_normalized  0
... category_normalized  0
... category_fr    0
... price_category  0
dtype: int64

Statistiques descriptives:
          title      price  rating  product_link ...  title_normalized  category_normalized  category_fr  price_category
y
count      1000  1000.0000  1000.0           1000  ...           1000  1000.0000  1000.0           1000
0
unique     999      NaN      5.0           1000  ...           999      50      50
3
3
top      The Star-Touched Queen      NaN      1.0  http://books.toscrape.com/catalogue/a-light-in...  ...  the star-touched queen      default      Défaut      Medi
m
freq       2      NaN     226.0           1  ...           2      152      152      60
6
mean      NaN  35.07935      NaN           NaN  ...           NaN      NaN      NaN      NaN
N
std       NaN  14.44669      NaN           NaN  ...           NaN      NaN      NaN      NaN
min      NaN  10.00000      NaN           NaN  ...           NaN      NaN      NaN      NaN
BooksToScrape_ETL > data_cleaning_analysis.py
... Codeium 105:21 LF UTF-8 4 spaces Python 3.12 (BooksToScrape_ETL) ⌂

```

```
BT BooksToScrape_ETL main
Project Run main
  G  E  C  : 
    N
    25%      NaN  22.10750  NaN
    50%      NaN  35.98000  NaN
    ... 75%      NaN  47.45750  NaN
    N
    max      NaN  59.99800  NaN
    N

[11 rows x 9 columns]

Nombre de livres par rating:
rating
1    226
2    196
3    203
4    179
5    196
Name: count, dtype: int64
D  ↴
  Prix moyen par rating:
rating_int
1    34.561195
2    34.819918
3    34.692020
4    36.093296
5    35.376498
Name: price, dtype: float64
I  ↴
  Nombre de livres par catégorie de prix:
price_category
  Medium   606
  High     198
  Low      196
Name: count, dtype: int64
...
  Nombre de livres par catégorie:
category
  default      152
  nonfiction   118
  sequential art  75
  add a comment  67
  fiction       65
  young adult   54
  fantasy        48
  romance        35
  mystery        32
  food and drink  30
  childrens      29
  historical fiction  26
D  ↴
  poetry        19
  classics       19
  history        18
  horror         17
  ↴
  womens fiction  17
  science fiction  16
  science          14
  ↴
  music           13
  business         12
  travel           11
  thriller          11
  ↴
  philosophy       11

BooksToScrape_ETL > data_cleaning_analysis.py
... Codeium 105:21 LF UTF-8 4 spaces Python 3.12 (BooksToScrape_ETL) ⌂
```

```
BT BooksToScrape_ETL main
Project Run main
  G  E  C  : 
    price_category
    Medium   606
    High     198
    Low      196
    Name: count, dtype: int64
...
    Nombre de livres par catégorie:
category
  default      152
  nonfiction   118
  sequential art  75
  add a comment  67
  fiction       65
  young adult   54
  fantasy        48
  romance        35
  mystery        32
  food and drink  30
  childrens      29
  historical fiction  26
D  ↴
  poetry        19
  classics       19
  history        18
  horror         17
  ↴
  womens fiction  17
  science fiction  16
  science          14
  ↴
  music           13
  business         12
  travel           11
  thriller          11
  ↴
  philosophy       11

BooksToScrape_ETL > data_cleaning_analysis.py
... Codeium 105:21 LF UTF-8 4 spaces Python 3.12 (BooksToScrape_ETL) ⌂
```

The screenshot shows a Jupyter Notebook interface with a dark theme. The top navigation bar includes tabs for 'main' and other files like 'scraping.py', 'data_cleaning_analysis.py', and 'data_visualization.py'. The main content area displays a Pandas DataFrame:

```
humor          10
autobiography     9
art             8
psychology        7
religion          7
spirituality       6
christian fiction   6
new adult          6
self help           5
sports and games      5
biography           5
health              4
christian            3
politics              3
contemporary         3
historical            2
parenting              1
paranormal              1
short stories          1
novels              1
crime                1
suspense              1
cultural              1
erotica                1
adult fiction          1
academic                1
Name: count, dtype: int64
```

Below the DataFrame, a note says 'Nombre de Livres par catégorie (en français):' followed by a table:

category_fr	Nombre de Livres
Défaut	152
Non-fiction	110

The bottom status bar indicates the code editor is 'Codeium', the file is 'main.py', and the Python version is 'Python 3.12 (BooksToScrape_ETL)'.

This screenshot shows the same Jupyter Notebook interface with a dark theme. The top navigation bar includes tabs for 'main' and other files. The main content area displays a Pandas DataFrame:

```
Art Séquentiel      75
Ajouter un commentaire    67
Fiction            65
Jeunes Adultes      54
Fantaisie           48
Romance             35
Mystère             32
Cuisine et Boissons 30
Pour Enfants         29
Fiction Historique  26
Classiques           19
Poésie               19
Histoire             18
Fiction Féminine    17
Horreur              17
Science-fiction     16
Science                14
Musique                13
Affaires              12
Philosophie           11
Thriller              11
Voyage                11
Humour                10
Autobiographie        9
Art                  8
Religion                7
Psychologie           7
Fiction Chrétienne    6
Nouveaux Adultes      6
Spiritualité           6
Biographie              5
Sports et Jeux          5
```

The bottom status bar indicates the code editor is 'Codeium', the file is 'main.py', and the Python version is 'Python 3.12 (BooksToScrape_ETL)'.

```
Développement Personnel      5
Santé                         4
Politique                      3
Chrétien                       3
Contemporain                   3
... Historique                  2
Suspense                       1
Académique                     1
Romans                         1
Parentalité                     1
Paranormal                      1
Nouvelles                      1
Fiction pour adultes          1
Culturel                        1
Crime                           1
Érotisme                        1
Name: count, dtype: int64

Les données nettoyées ont été sauvegardées dans output/books_clean.csv
```

```
Data Cleaning and Analysis completed successfully.
```

```
Starting Data Visualization...
```

```
Données chargées pour visualisation depuis output/books_clean.csv
```

```
Data Visualization completed successfully.
```

```
Full ETL Pipeline executed successfully!
```

💡 Ces captures attestent du bon fonctionnement du pipeline ETL, depuis l'extraction des données jusqu'à leur analyse finale.

8.6. Conclusion de l'Annexe

Cette annexe permet d'accéder aux versions complètes des codes, aux visualisations supplémentaires et aux explications techniques détaillées.

Elle constitue une référence approfondie pour ceux qui souhaitent analyser le projet dans son intégralité.