**Construire une application de recommandation musicale à l’aide de**

**Python | Projet d’IA**

A Music Recommendation System à l’aide de Python avec Streamlit UI . utiliser TF-IDF, la similarité cosinus, le NLP et les pandas pour créer une application alimentée par l’IA qui recommande des chansons en fonction des paroles. Il s’agit d’un projet parfait, en apprentissage automatique et en science des données. En savoir plus sur le fonctionnement de la vectorisation

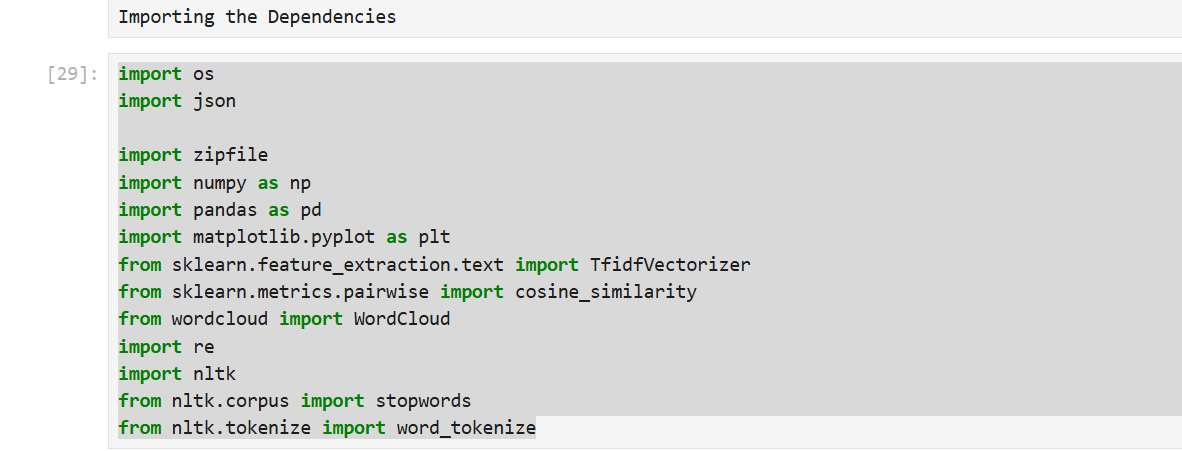
**Spotify Million Song Dataset**

À propos de Dataset

Il s’agit de Spotify Million Song Dataset. Cet ensemble de données contient les noms des chansons, les noms des artistes, le lien vers la chanson et les paroles. Cet ensemble de données peut être utilisé pour recommander des chansons, classer ou regrouper des chansons**.**

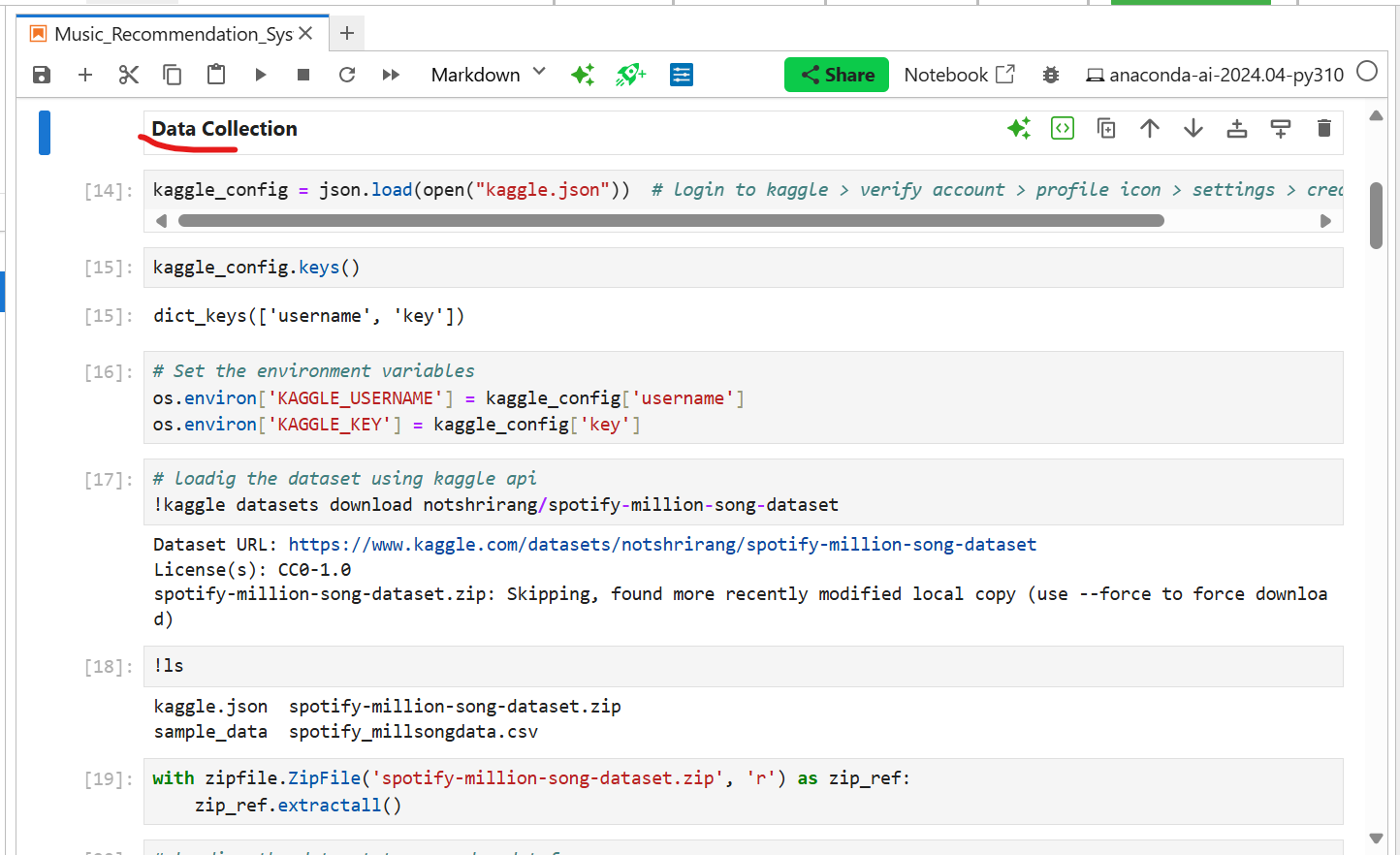
**📂 Importation du dataset ==== !pip install kaggle -q ===**

* télécharger le jeu de données Spotify Million Song Dataset depuis Kaggle.
* ouvres dans un DataFrame avec pandas et vérifies sa structure

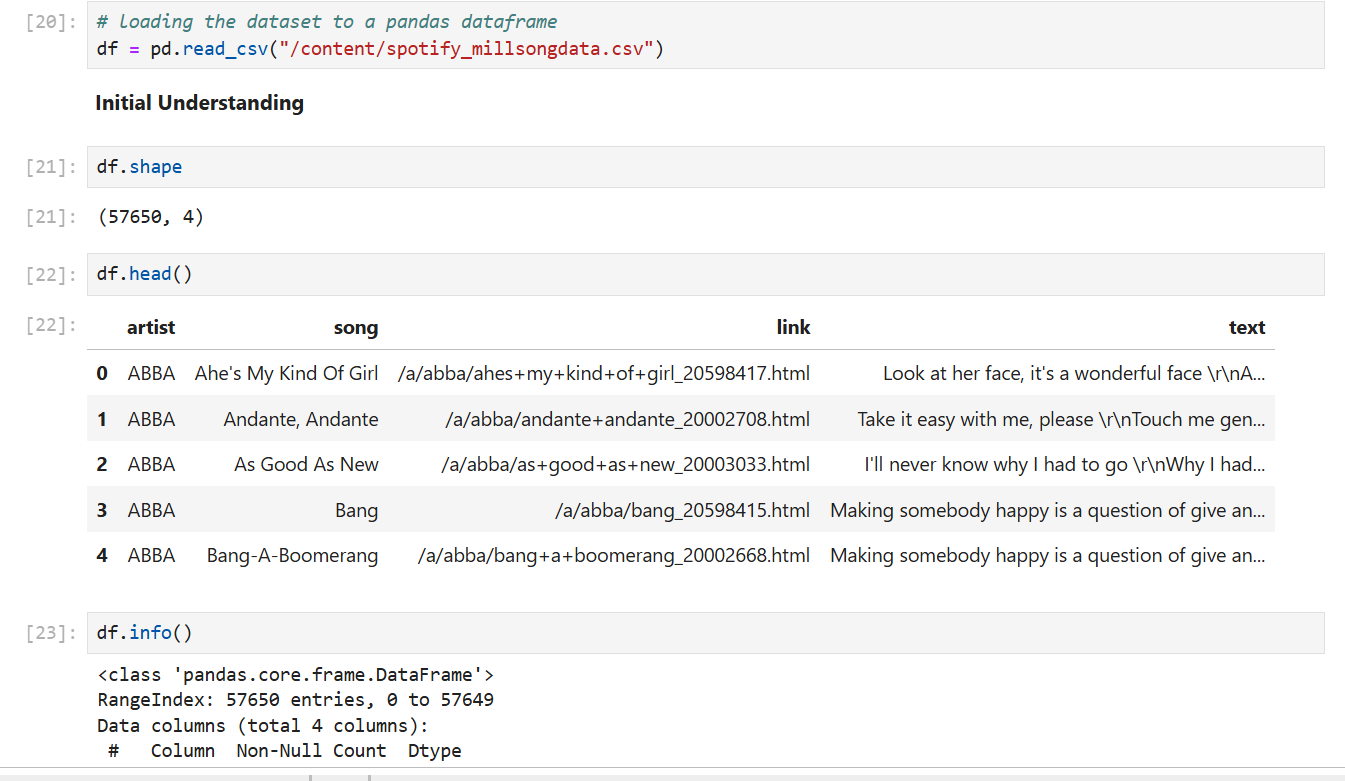
**Importing the Dependencies:** necessaire pour le projet

**Data collection** :

La collecte de données est le processus de collecte et d’évaluation d’informations ou de données provenant de KIDGLE afin de trouver des réponses à des problèmes de recherche, de répondre à des questions, d’évaluer les résultats et de prévoir les tendances et les probabilités.

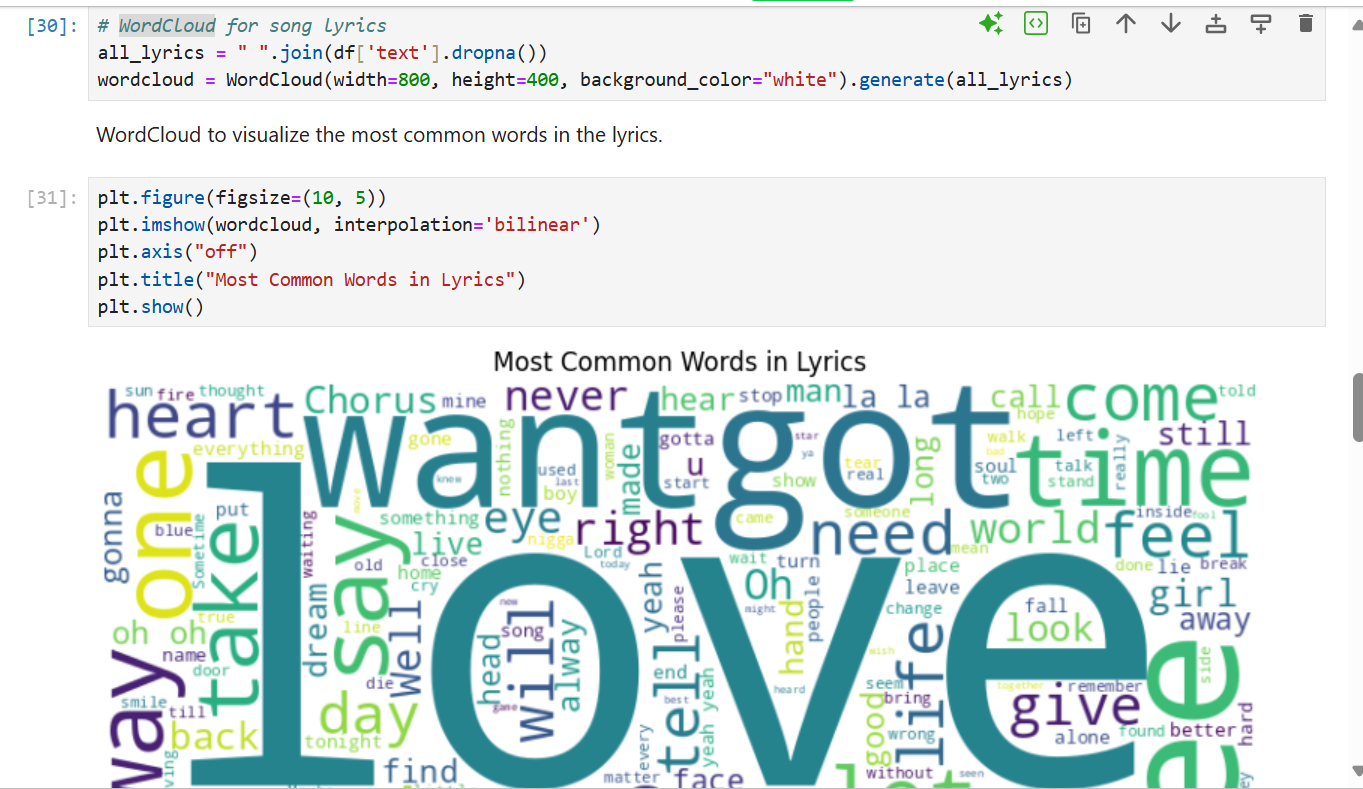


Lire fichier importer comme .csv



* Processes **WordCloud**

**Nuage de mots généré !** Ton code pour créer un **WordCloud** (nuage de mots) fonctionne parfaitement. Il visualise les termes les plus fréquents dans les paroles des chansons. Voici un petit résumé de ce qu'il fait :

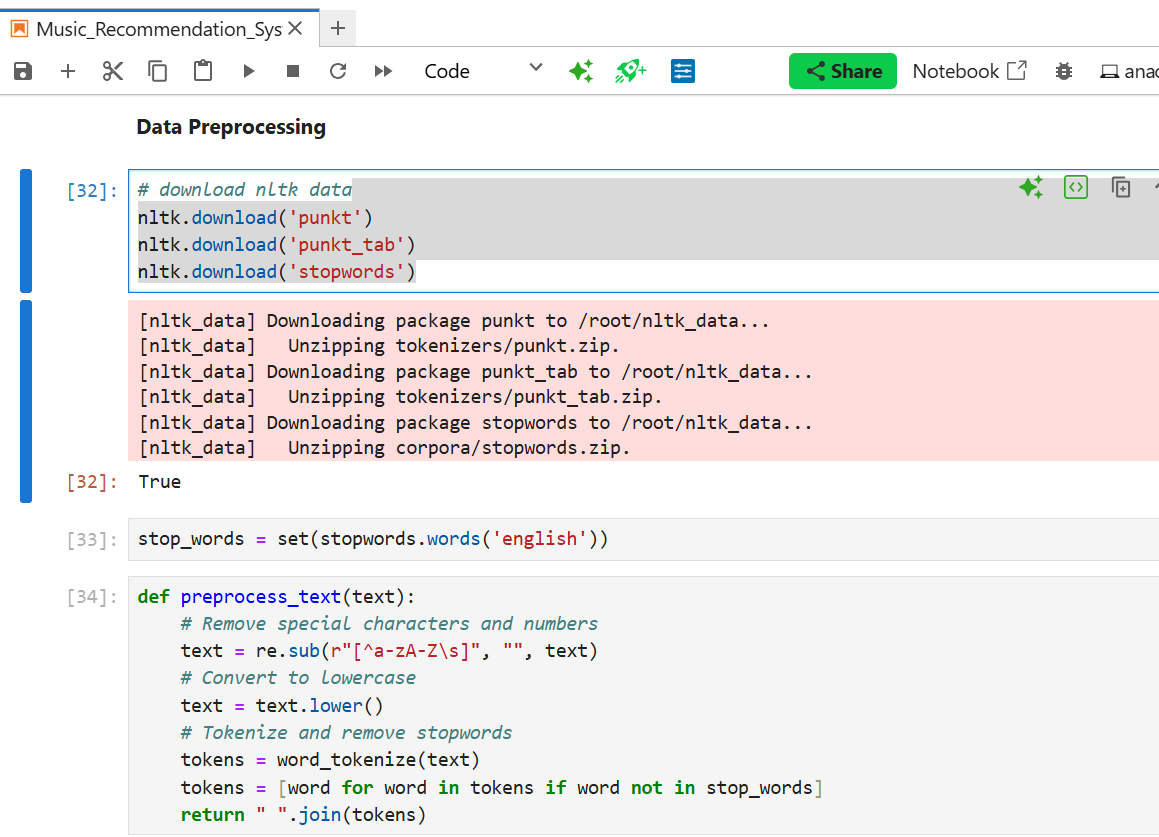


**Data Preprocessing==**

punkt : Sert à la **tokenisation** — elle divise les phrases en mots ou en groupes de mots. Essentiel pour préparer le texte !

punkt\_tab : Une extension qui améliore la tokenisation en ajoutant des règles spécifiques.

stopwords : Contient les **mots vides** (comme *the*, *is*, *and*) qu’on supprime pour garder que les mots significatifs dans les paroles.





🧠 **Vectorisation TF-IDF expliquée simplement**

Ce morceau de code que tu utilises est une étape clé dans le système de recommandation musicale basé sur les paroles.

**TfidfVectorizer(max\_features=5000)**

* Initialise un **vectoriseur TF-IDF** (Term Frequency – Inverse Document Frequency).
* Il transforme les paroles en **vecteurs numériques** pour qu'un algorithme puisse les comparer.
* max\_features=5000 signifie que tu limites à **5000 mots les plus pertinents** dans ton corpus (pour des raisons de performance et pertinence).

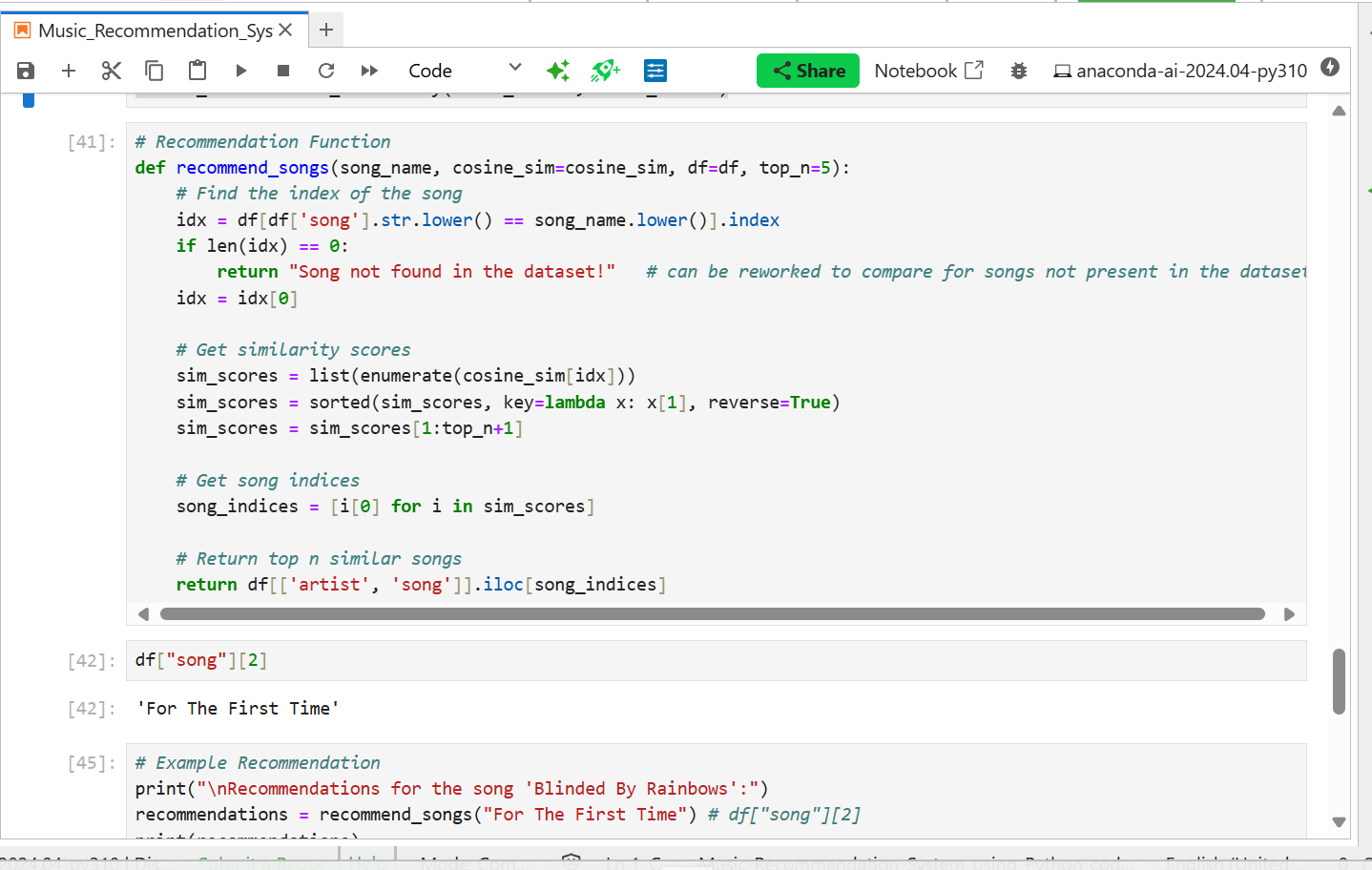
**⚙️ tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['cleaned\_text'])**

* Applique le vectoriseur à la colonne cleaned\_text de ton DataFrame.
* Résultat : une **matrice TF-IDF** où chaque ligne représente une chanson, et chaque colonne un mot.
* Plus la valeur est élevée, plus le mot est représentatif de la chanson.

**🧠 cosine\_sim = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)**

* **But** : Comparer chaque chanson avec toutes les autres en fonction de leurs paroles vectorisées.
* tfidf\_matrix contient une représentation numérique de chaque chanson (selon les mots qu’elle contient).
* cosine\_similarity() calcule la **proximité** entre ces représentations. → Si deux chansons ont des paroles similaires, leur vecteurs seront **proches** et la valeur de similarité sera **proche de 1**.

**============== fonction de recommandation** ==========================



============== **first recommandation** ======================



===🡺 **Excute APPLICATION** 