SAE S5C.01 : Proposer une solution optimisée à partir de données internes et externe



**Robin FOEX, Nabih EL-HOUB, Tom Lesellier, Milhane Rabehi**

[Abstract 4](#_Toc190459981)

[Analyse du modèle OLTP et optimisations mises en place 4](#_Toc190459982)

[Modélisation relationnelle OLTP 4](#_Toc190459983)

[Structure et Organisation 5](#_Toc190459984)

[TABLE TITLE (Films et séries) 5](#_Toc190459985)

[TABLE NAMES (Acteurs, réalisateurs, etc.) 5](#_Toc190459986)

[TABLE COMMENTARY (Critiques des utilisateurs) 5](#_Toc190459987)

[TABLE GENRE (Genres de films) 5](#_Toc190459988)

[TABLE LISTED (Association entre films et genres) 5](#_Toc190459989)

[TABLE PRINCIPAL (Relations entre films et personnes) 6](#_Toc190459990)

[TABLE CATEGORY (Catégories de rôles) 6](#_Toc190459991)

[TABLE EPISODE (Épisodes des séries) 6](#_Toc190459992)

[TABLE ALTERNATETITLE (Titres alternatifs) 6](#_Toc190459993)

[TABLE PROFESSION (Professions des personnes) 6](#_Toc190459994)

[TABLE BE (Lien entre personnes et professions) 6](#_Toc190459995)

[TABLE LANGUAGE (Langues des films et séries) 6](#_Toc190459996)

[TABLE REGION (Régions associées aux films) 7](#_Toc190459997)

[TABLE TYPE (Types de films ou séries*)* 7](#_Toc190459998)

[TABLE TYPETITLEALTERNATE (Lien entre types et titres) 7](#_Toc190459999)

[Clés primaires et normalisation 7](#_Toc190460000)

[Objectif 7](#_Toc190460001)

[Indexation pour optimiser les performances 8](#_Toc190460002)

[Taille de la base OLTP 10](#_Toc190460003)

[Partitionnement des tables volumineuses 10](#_Toc190460004)

[Conclusion 10](#_Toc190460005)

[Analyse du modèle en étoile 10](#_Toc190460006)

[Structure du modèle en étoile et choix 11](#_Toc190460007)

[Objectif et utilisation 11](#_Toc190460008)

[Tables de dimensions 12](#_Toc190460009)

[Table T\_DIM\_COMMENTARY (Dimension des commentaires) 12](#_Toc190460010)

[Table T\_DIM\_GENRE (Dimension des genres de films) 12](#_Toc190460011)

[Table T\_DIM\_NAMES (Dimension des personnes) 12](#_Toc190460012)

[Table T\_DIM\_ROLE (Dimension des rôles) 12](#_Toc190460013)

[Table T\_DIM\_TITLE (Dimension des films et séries) 12](#_Toc190460014)

[Tables de faits (mesures et statistiques) 13](#_Toc190460015)

[Table T\_FACT\_KEYWORD (Faits liés aux mots-clés des films et séries) 13](#_Toc190460016)

[Table T\_FACT\_MEDIA (Faits liés aux évaluations et popularité des films) 13](#_Toc190460017)

[Taille de la base OLAP 13](#_Toc190460018)

[Importation des données 14](#_Toc190460019)

[Export IMDB 14](#_Toc190460020)

[Transformation avec Python des CSV de la base IMDB 14](#_Toc190460021)

[Importation des données dans la base OLTP 14](#_Toc190460022)

[Migration des données vers la Base OLAP 16](#_Toc190460023)

[Rapport BI 19](#_Toc190460024)

[Modèles IA produits et démarche de développement et d’optimisation 21](#_Toc190460025)

[Données d’apprentissage 21](#_Toc190460026)

[Prétraitement des données 21](#_Toc190460027)

[Types de modèles utilisés 21](#_Toc190460028)

[Modèle 1 : Classification binaire du sentiment 21](#_Toc190460029)

[Modèle 2 : Attribution automatique d’une note (1 à 10) 23](#_Toc190460030)

[Modèle 3 : Score gradué du sentiment 26](#_Toc190460031)

# Abstract

L'Internet Movie Database (IMDb) est une base de données de référence sur le cinéma, la télévision et les jeux vidéo. Notre projet vise à concevoir une architecture optimisée pour les données d’IMDb, tout en exploitant des modèles d’intelligence artificielle (IA) pour l’analyse des critiques de films.

La première phase consistera à structurer une base de données relationnelle OLTP sur SQL Server à partir des fichiers plats, en appliquant des techniques d’indexation et de partitionnement. L'importation des données sera réalisée via Python.

Ensuite, des rapports Power BI seront mis en place pour permettre une visualisation efficace des données, avec l'intégration d’un modèle OLAP pour une analyse approfondie. La deuxième phase du projet consistera à développer plusieurs modèles d’IA en Python.

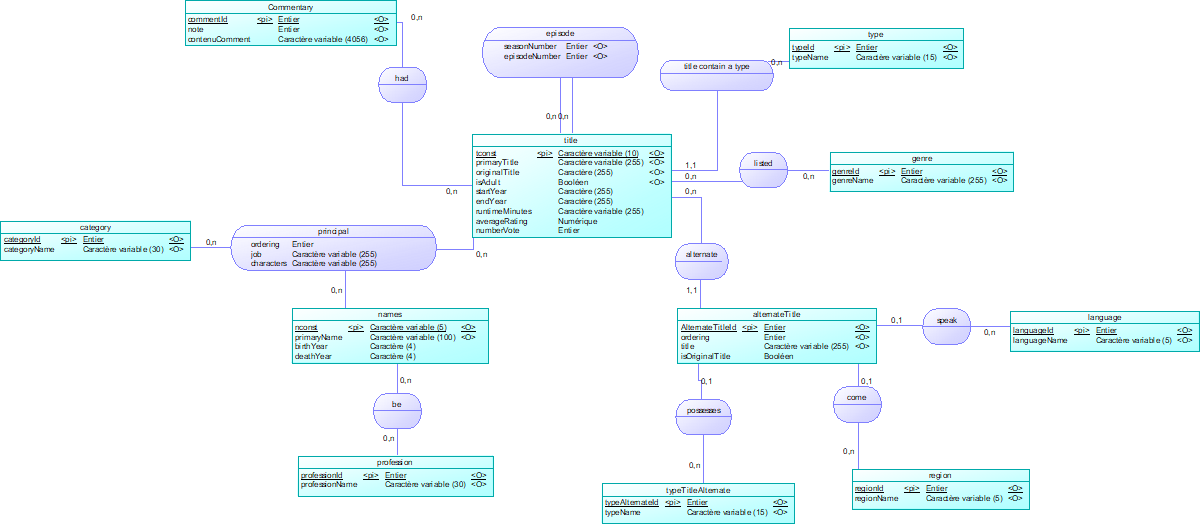
Le premier modèle déterminera automatiquement la polarité (positive ou négative) des critiques en s'appuyant sur un dataset existant de Stanford. Le second attribuera une note sur 10 à chaque critique afin de renforcer la cohérence des évaluations. Enfin, le dernier modèle proposera un score gradué entre l’évaluation binaire et la notation sur 10 niveaux.

Ces modèles seront conçus pour fonctionner efficacement tout en minimisant la consommation de ressources. Notre projet allie gestion de bases de données et intelligence artificielle afin d’optimiser le traitement et l’exploitation des critiques de films sur IMDb

# Analyse du modèle OLTP et optimisations mises en place

Notre modèle OLTP de la base de données d’IMDb est structuré autour de plusieurs tables représentant les entités essentielles de la base. Les optimisations ont été mises en place à travers principalement de l’indexation et la gestion des clés primaires.

## Modélisation relationnelle OLTP

Ce modèle relationnel est structuré en plusieurs tables interconnectées pour minimiser la redondance et optimiser la performance des requêtes. Voici les principales tables :

La base de données OLTP d’IMDb est conçue pour **stocker et gérer efficacement** les informations liées aux films, séries, acteurs, réalisateurs, genres et autres éléments essentiels de l'industrie cinématographique. Ce modèle relationnel structuré permet **d’assurer la cohérence et l’intégrité** des données en normalisant les informations et en évitant la redondance.

## Structure et Organisation

La base OLTP suit un modèle relationnel divisé en plusieurs catégories d’entités, chacune représentée par une table. Ces entités sont organisées de manière à faciliter l’accès et la mise à jour des données, tout en assurant une traçabilité optimale. Voici les principales catégories de données contenues dans cette base :

### TABLE TITLE (Films et séries)

Stocke les informations essentielles sur les films et séries répertoriés sur IMDb.

Clé primaire : TCONST (identifiant unique du film ou de la série)

Colonnes principales :

PRIMARYTITLE : Titre principal du film/série

ORIGINALTITLE : Titre original

ISADULT : Indique si le contenu est pour adultes

STARTYEAR / ENDYEAR : Années de début et fin (pour les séries)

RUNTIME : Durée en minutes

AVERAGERATING : Note moyenne attribuée par les utilisateurs

NUMBERVOTE : Nombre total de votes

### TABLE NAMES (Acteurs, réalisateurs, etc.)

Stocke les informations sur les personnes impliquées dans les films et séries.

Clé primaire : NCONST (identifiant unique de la personne)

Colonnes principales :

PRIMARYNAME : Nom principal

BIRTHYEAR / DEATHYEAR : Années de naissance et décès

KNOWNFORTITLES : Liste des films pour lesquels la personne est connue

### TABLE COMMENTARY (Critiques des utilisateurs)

Stocke les avis des spectateurs sur les films et séries.

Clé primaire : COMMENTID

Colonnes principales :

TCONST : Référence au film ou série concerné

NCONST : Référence à l’utilisateur ayant laissé la critique

COMMENTTEXT : Texte de la critique

NOTE : Note attribuée (sur 10)

DATECOMMENT : Date de publication

### TABLE GENRE (Genres de films)

Liste des genres disponibles dans IMDb.

Clé primaire : GENREID

Colonnes principales :

GENRENAME : Nom du genre (Action, Drame, Comédie, etc.)

### TABLE LISTED (Association entre films et genres)

Permet de lier un film à plusieurs genres.

Clé primaire : (GENREID, TCONST)

Colonnes principales :

TCONST : Film ou série associé

GENREID : Genre correspondant

### TABLE PRINCIPAL (Relations entre films et personnes)

Permet d’identifier les rôles des personnes associées aux films et séries.

Clé primaire : (NCONST, TCONST, CATEGORYID, ORDERING)

Colonnes principales :

TCONST : Référence au film ou série

NCONST : Référence à la personne

CATEGORYID : Catégorie du rôle (acteur, réalisateur, scénariste, etc.)

CHARACTERS : Personnage interprété (si applicable)

ORDERING : Ordre d’apparition dans le générique

### TABLE CATEGORY (Catégories de rôles)

Stocke les différentes catégories de participation aux films/séries.

Clé primaire : CATEGORYID

Colonnes principales :

CATEGORYNAME : Nom de la catégorie (acteur, réalisateur, producteur, etc.)

### TABLE EPISODE (Épisodes des séries)

Stocke les épisodes de chaque série.

Clé primaire : (TCONST, TIT\_TCONST)

Colonnes principales :

TCONST : Identifiant de l’épisode

TIT\_TCONST : Référence à la série principale

SEASONNUMBER : Numéro de la saison

EPISODENUMBER : Numéro de l’épisode

### TABLE ALTERNATETITLE (Titres alternatifs)

Stocke les titres alternatifs des films et séries.

Clé primaire : ALTERNATETITLEID

Colonnes principales :

TCONST : Référence au film ou série

TITLE : Titre alternatif

ISORIGINALTITLE : Indique si ce titre est l’original

### TABLE PROFESSION (Professions des personnes)

Stocke les différents types de professions liées aux films et séries.

Clé primaire : PROFESSIONID

Colonnes principales :

PROFESSIONNAME : Nom de la profession (Acteur, Réalisateur, Monteur, etc.)

### TABLE BE (Lien entre personnes et professions)

Permet d’associer une personne à une ou plusieurs professions.

Clé primaire : (NCONST, PROFESSIONID)

Colonnes principales :

NCONST : Identifiant de la personne

PROFESSIONID : Référence à la profession

### TABLE LANGUAGE (Langues des films et séries)

Stocke les langues utilisées dans les productions IMDb.

Clé primaire : LANGUAGEID

Colonnes principales :

LANGUAGENAME : Nom de la langue

### TABLE REGION (Régions associées aux films)

Stocke les informations sur les pays ou régions où un film est distribué.

Clé primaire : REGIONID

Colonnes principales :

REGIONNAME : Nom de la région

### TABLE TYPE (Types de films ou séries*)*

Stocke les différents types de contenus présents sur IMDb.

Clé primaire : TYPEID

Colonnes principales :

TYPENAME : Nom du type (Film, Série, Court-métrage, etc.)

### TABLE TYPETITLEALTERNATE (Lien entre types et titres)

Stocke les associations entre les films et leurs types respectifs.

Clé primaire : TYPEALTERNATEID

Colonnes principales :

TCONST : Référence au film/série

TYPEID : Référence au type

## Clés primaires et normalisation

Chaque table possède une clé primaire définie pour assurer l’unicité des enregistrements et adapter les recherches sur les identifiants. Par exemple :

PK\_TITLE (TCONST) : Identifiant unique pour chaque film ou série.

PK\_NAMES (NCONST) : Identifiant unique pour chaque personne.

PK\_COMMENTARY (COMMENTID) : Identifiant unique pour chaque critique.

Cela permet de limiter la redondance des données et améliorer la cohérence des mises à jour.

## Objectif

On a énormément de données et créer des index dans notre base de données ont plusieurs objectifs essentiels :

* **L’amélioration des recherches et des filtrages**

Les index accélèrent considérablement les recherches lorsqu'on filtre des données sur des colonnes spécifiques. Par exemple :

IDX\_ALTERNATETITLE\_TITLE améliore la rapidité des recherches sur les titres alternatifs des films et séries.

IDX\_NAMES\_PRIMARYNAME facilite la recherche des personnes (acteurs, réalisateurs) par leur nom.

* **L’optimisation des jointures entre les tables**

Certains index sont placés sur des colonnes servant à relier plusieurs tables entre elles, permettant ainsi des jointures plus rapides et plus efficaces. Par exemple :

IDX\_FK\_ALTERNAT\_ALTERNATE\_TITLE accélère les associations entre la table ALTERNATETITLE et TITLE.

IDX\_FK\_LISTED\_LISTED\_TITLE améliore la correspondance entre les films/séries et leurs genres.

IDX\_FK\_PRINCIPA\_PRINCIPAL\_NAMES accélère les recherches sur les rôles des acteurs et réalisateurs.

* **L’accélération des tris et regroupements**

Certains index permettent d'optimiser le tri des données et les opérations de regroupement (GROUP BY, ORDER BY).

IDX\_TITLE\_AVERAGERATING facilite l'accès aux films en fonction de leur note moyenne.

IDX\_TITLE\_NUMBERVOTE accélère l'affichage des films les plus populaires.

IDX\_EPISODE\_SEASONNUMBER permet un tri plus rapide des épisodes par saison.

* **Réduire le temps d’exécution des requêtes**

Grâce à ces index, les bases de données évitent de parcourir entièrement les tables et accèdent directement aux lignes nécessaires, rendant les requêtes beaucoup plus rapides.

Ainsi, les index sont des éléments clés qui garantissent des performances optimales pour les recherches, les jointures et les tris au sein de notre base de données.

## Indexation pour optimiser les performances

Table : ALTERNATETITLE

create index IDX\_FK\_ALTERNAT\_ALTERNATE\_TITLE on ALTERNATETITLE (TCONST);

create index IDX\_FK\_ALTERNAT\_COME\_REGION on ALTERNATETITLE (REGIONID);

create index IDX\_FK\_ALTERNAT\_POSSESSES\_TYPETITL on ALTERNATETITLE (TYPEALTERNATEID);

create index IDX\_FK\_ALTERNAT\_SPEAK\_LANGUAGE on ALTERNATETITLE (LANGUAGEID);

create index IDX\_ALTERNATETITLE\_TITLE on ALTERNATETITLE (TITLE);

create index IDX\_ALTERNATETITLE\_ISORIGINALTITLE on ALTERNATETITLE (ISORIGINALTITLE);

Table : BE

create index IDX\_FK\_BE\_BE\_PROFESSI on BE (PROFESSIONID);

create index IDX\_FK\_BE\_BE2\_NAMES on BE (NCONST);

Table : EPISODE

create index IDX\_FK\_EPISODE\_EPISODE\_TITLE on EPISODE (TIT\_TCONST);

create index IDX\_FK\_EPISODE\_EPISODE2\_TITLE on EPISODE (TCONST);

create index IDX\_EPISODE\_SEASONNUMBER on EPISODE (SEASONNUMBER);

create index IDX\_EPISODE\_EPISODENUMBER on EPISODE (EPISODENUMBER);

Table : HAD

create index IDX\_FK\_HAD\_HAD\_COMMENTA on HAD (COMMENTID);

create index IDX\_FK\_HAD\_HAD2\_TITLE on HAD (TCONST);

Table : LISTED

create index IDX\_FK\_LISTED\_LISTED\_TITLE on LISTED (TCONST);

create index IDX\_FK\_LISTED\_LISTED2\_GENRE on LISTED (GENREID);

Table : PRINCIPAL

create index IDX\_FK\_PRINCIPA\_PRINCIPAL\_CATEGORY on PRINCIPAL (CATEGORYID);

create index IDX\_FK\_PRINCIPA\_PRINCIPAL\_NAMES on PRINCIPAL (NCONST);

create index IDX\_FK\_PRINCIPA\_PRINCIPAL\_TITLE on PRINCIPAL (TCONST);

create index IDX\_PRINCIPAL\_TCONST on PRINCIPAL (TCONST);

create index IDX\_PRINCIPAL\_CATEGORYID on PRINCIPAL (CATEGORYID);

Table : TITLE

create index IDX\_FK\_TITLE\_TITLE\_CON\_TYPE on TITLE (TYPEID);

create index IDX\_TITLE\_PRIMARYTITLE on TITLE (PRIMARYTITLE);

create index IDX\_TITLE\_AVERAGERATING on TITLE (AVERAGERATING);

create index IDX\_TITLE\_NUMBERVOTE on TITLE (NUMBERVOTE);

Table : NAMES

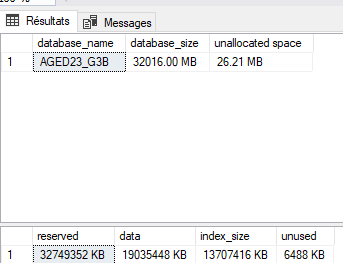
create index IDX\_NAMES\_PRIMARYNAME on NAMES (PRIMARYNAME);

create index IDX\_NAMES\_BIRTHYEAR on NAMES (BIRTHYEAR);

Table : COMMENTARY

create index IDX\_COMMENTARY\_NOTE on COMMENTARY (NOTE);

## Taille de la base OLTP



AGED23\_G3B

* Taille de la base : 32 016 Mo
* Espace réservé : 32 749 Mo
* Espace non alloué : 26,21 Mo
* Données : 18 150 Mo
* Index : 13 070 Mo
* Espace inutilisé : 6,34 Mo

### Partitionnement des tables volumineuses

Le partitionnement des tables peut être implémenté pour optimiser l’accès aux grandes quantités de données, notamment :

TITLE (films et séries)

COMMENTARY (critiques utilisateurs)

PRINCIPAL (relation film-personnes)

Par exemple, on peut partitionner la table COMMENTARY selon les années de publication des critiques.

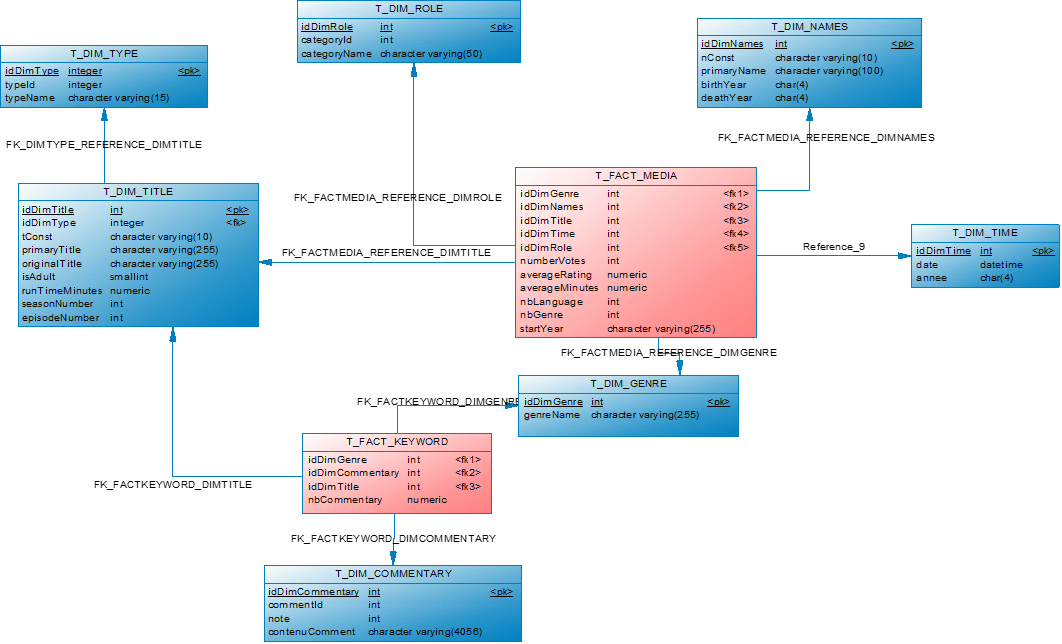
## Conclusion

Notre base de données OLTP a été conçue pour optimiser le stockage et la récupération des informations, en appliquant des principes de normalisation, des index et des clés primaires bien définies. Ces optimisations garantissent des performances intéressantes pour les transactions et les analyses effectuées sur la base.

# Analyse du modèle en étoile

Le modèle OLAP est structuré avec :

* Tables de dimensions : T\_DIM\_COMMENTARY, T\_DIM\_GENRE, T\_DIM\_NAMES, T\_DIM\_ROLE, T\_DIM\_TITLE
* Tables de faits : T\_FACT\_KEYWORD, T\_FACT\_MEDIA



## Structure du modèle en étoile et choix

Ce modèle en étoile utilisé dans la base OLAP IMDb est conçu pour optimiser l'analyse des données et les performances des requêtes. Il est composé de 2 tables de faits contenant les mesures chiffrées et de tables de dimensions apportant des informations contextuelles.

## Objectif et utilisation

La base OLAP IMDb est conçue pour **analyser et structurer** les données issues de la base OLTP afin de faciliter des analyses avancées sur les films, séries, critiques, genres et popularité des œuvres audiovisuelles.

Elle permet :

Une **agrégation rapide des données** pour produire des indicateurs clés.  
Une **analyse approfondie** des tendances cinématographiques.  
Une **optimisation des requêtes** pour des rapports de Business Intelligence.

La base suit un **modèle en étoile**, avec des **tables de dimensions** et **des tables de faits**, facilitant ainsi la navigation et l’exploration des données.

**Séparation des dimensions et des faits :**

Optimisation des requêtes : Les données sont organisées pour accélérer les requêtes analytiques.

Évolutivité : Facilement extensible en ajoutant de nouvelles dimensions ou de nouveaux faits.

Meilleure lisibilité : Les analystes peuvent comprendre et exploiter les données sans complexité.

**Stockage des données agrégées :**

Permet d'éviter de recalculer les statistiques à chaque requête.

Améliore la performance en évitant les jointures complexes sur les grandes bases de données.

## Tables de dimensions

### Table T\_DIM\_COMMENTARY (Dimension des commentaires)

Cette table contient des informations détaillées sur les commentaires des utilisateurs. Elle permet l'analyse des tendances et des sentiments sur les films.

Clé primaire : **IDDIMCOMMENTARY** (identifiant unique de la dimension critique)

Colonnes principales :

* **COMMENTID** : Identifiant unique de la table COMMENTARY de la base OLTP
* **NOTE** : Note attribuée par l’utilisateur (échelle de 1 à 10)
* **CONTENUCOMMENT** : Texte de la critique
* **DATECOMMENT** : Date de publication de la critique

### Table T\_DIM\_GENRE (Dimension des genres de films)

Cette table liste tous les genres cinématographiques existants et leurs caractéristiques.

Clé primaire : **IDDIMGENRE** (identifiant unique du genre)

Colonnes principales :

* **GENREID** : Identifiant du genre
* **GENRENAME** : Nom du genre (Action, Drame, Comédie, Science-fiction, etc.)

### Table T\_DIM\_NAMES (Dimension des personnes)

Cette table contient les informations sur les acteurs, réalisateurs et autres contributeurs à un film ou une série.

Clé primaire : **IDDIMNAMES** (identifiant unique de la personne)

Colonnes principales :

* **NCONST** : Identifiant unique de la personne
* **PRIMARYNAME** : Nom complet de la personne
* **BIRTHYEAR / DEATHYEAR** : Années de naissance et décès (si applicable)
* **KNOWNFORTITLES** : Liste des films ou séries pour lesquels cette personne est connue

### Table T\_DIM\_ROLE (Dimension des rôles)

Cette table catégorise les différents rôles joués par les personnes dans les films et séries.

Clé primaire : **IDDIMROLE** (identifiant unique du rôle)

Colonnes principales :

* **CATEGORYID** : Identifiant du type de rôle
* **CATEGORYNAME** : Nom du rôle (acteur principal, réalisateur, producteur, scénariste, etc.)

### Table T\_DIM\_TITLE (Dimension des films et séries)

Cette table contient les informations générales sur les films et séries

Clé primaire : **IDDIMTITLE** (identifiant unique du film ou de la série)

Colonnes principales :

* **TCONST** : Identifiant unique du film ou de la série
* **PRIMARYTITLE** : Titre principal
* **ORIGINALTITLE** : Titre original
* **ISADULT** : Indique si le contenu est destiné aux adultes
* **STARTYEAR** / **ENDYEAR** : Années de sortie et de fin (si applicable)
* **RUNTIME** : Durée du film/série en minutes

## Tables de faits (mesures et statistiques)

### Table T\_FACT\_KEYWORD (Faits liés aux mots-clés des films et séries)

Cette table stocke des informations sur les mots-clés associés aux films et séries. Elle est utile pour l’analyse des tendances et des thèmes les plus populaires.

Clé primaire : **IDFACTKEYWORD**

Clés étrangères :

* **IDDIMTITLE** (lien avec T\_DIM\_TITLE)

Colonnes principales :

* **KEYWORD** : Mot-clé lié au film ou à la série
* **FREQUENCY** : Nombre d’occurrences du mot-clé dans les descriptions et critiques

### Table T\_FACT\_MEDIA (Faits liés aux évaluations et popularité des films)

Cette table contient des mesures statistiques sur les films et séries.

Clé primaire : **IDFACTMEDIA**

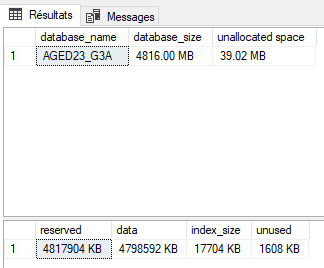
Clés étrangères :

* **IDDIMTITLE** (lien avec T\_DIM\_TITLE)
* **IDDIMCOMMENTARY** (lien avec T\_DIM\_COMMENTARY)
* **IDDIMGENRE** (lien avec T\_DIM\_GENRE)

Colonnes principales :

* **NUMBERVOTES** : Nombre de votes attribués
* **AVERAGERATING** : Moyenne des notes données par les utilisateurs
* **TOTALCOMMENTS** : Nombre total de critiques

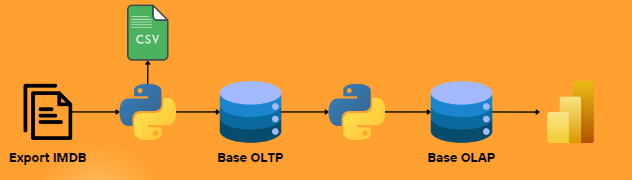
## Taille de la base OLAP



AGED23\_G3A

* Taille de la base : 4 816 Mo
* Espace réservé : 4 817 Mo
* Espace non alloué : 39,02 Mo
* Données : 4 798 Mo
* Index : 17 Mo
* Espace inutilisé : 1,6 Mo

# Importation des données



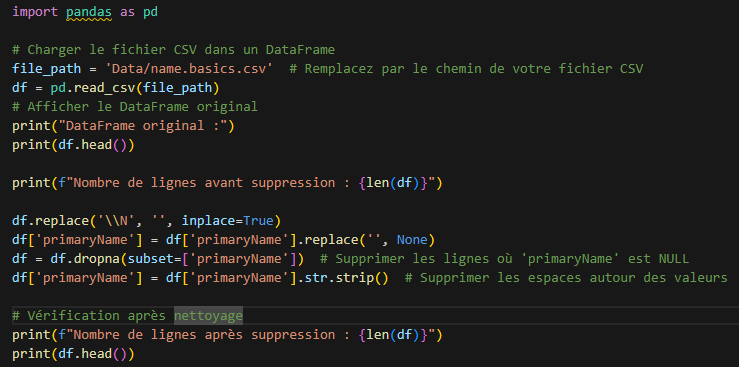
L’importation des données que ça soit pour le modèle OLTP ou OLAP ont été réaliser en script python. Car nous avons déjà utilisé ce langage en données massives, technique d’IA pour réaliser des exports, data frames, etc.

## Export IMDB

Extraction des données brutes de la base IMDB sous forme de fichiers CSV, comme “name.basics.csv”

## Transformation avec Python des CSV de la base IMDB

Une fois les données des CSV importés, dans python, il faut les nettoyer pour qu’elles soient conformes aux tables présentent dans la base OLTP. Pour cela on a créé des data frames à l’aide du package pandas

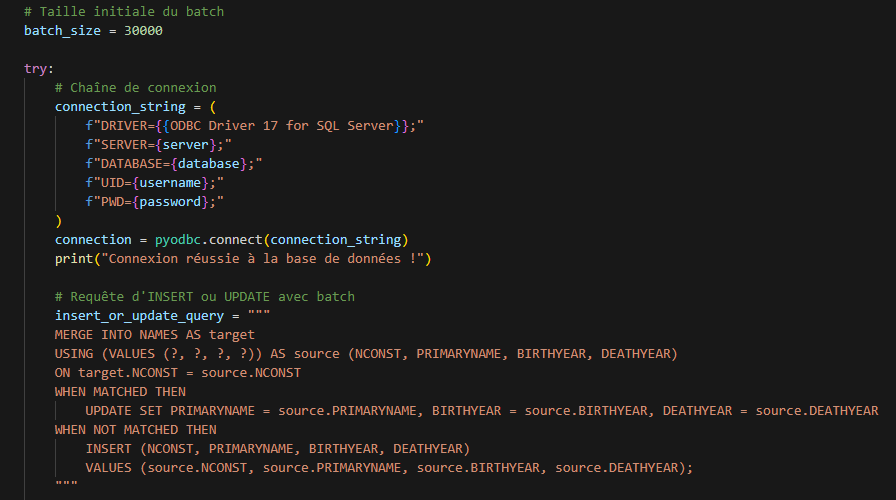
:  


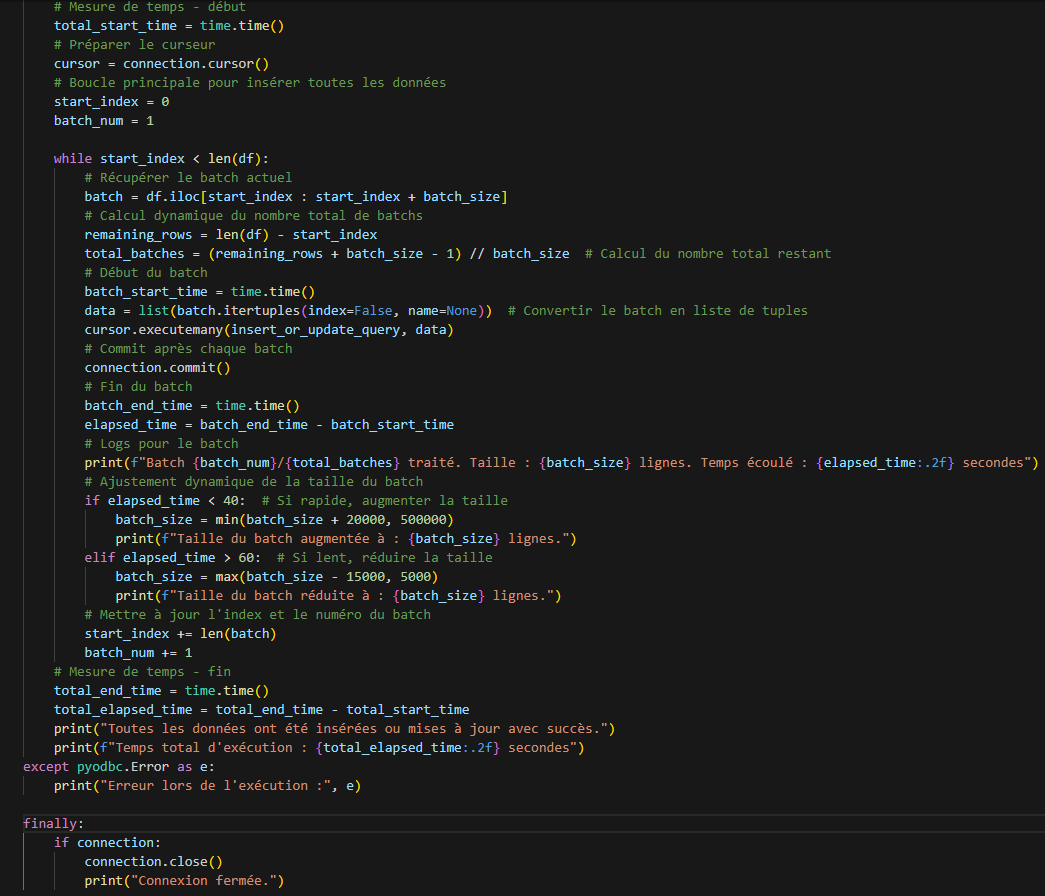
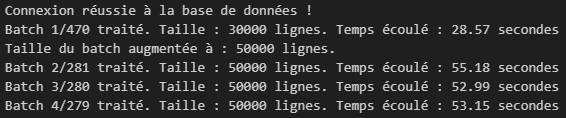
Ce code supprime les lignes où le champ du nom de la personne est vide, puis affiche le DataFrame afin de vérifier que les modifications ont bien été appliquées.

Une fois toutes les corrections effectuées sur chaque fichier CSV et que les DataFrames sont conformes aux tables de la base OLTP, l'importation des données dans la base peut être réalisée.

## Importation des données dans la base OLTP

L'importation des données est principalement effectuée en batch afin d'insérer plusieurs lignes simultanément à l'aide de la fonction "executemany". Cette méthode permet d'optimiser les performances et de réduire le temps d'exécution lors de l'insertion des données dans la base de données.

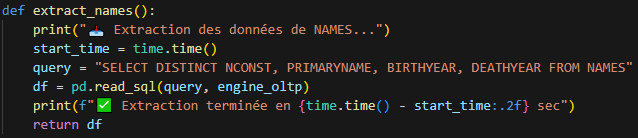


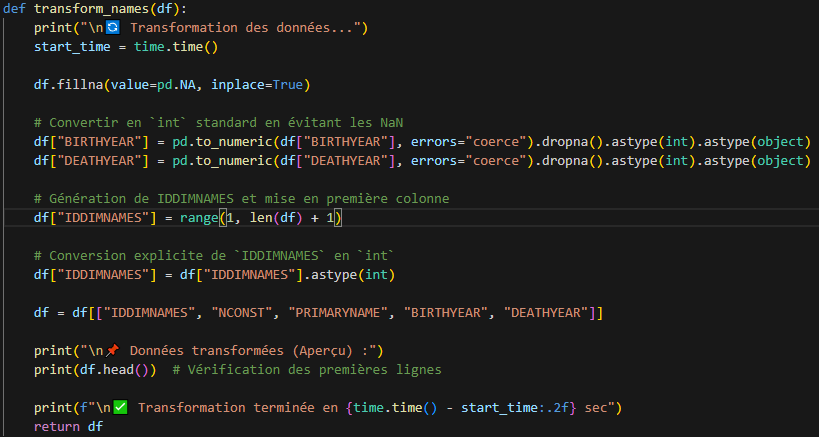
Dans le code, nous avons réalisé un algorithme qui effectue un ajustement au niveau de la taille du batch. En fonction de la durée de l’insertion du batch, soit il baisse ou augmente.  
Voici un exemple d’insertion :  


Une fois toute les tables de la base OLTP inséré, on passe à l’ETL qui permet de récupérer les données de la base OLTP pour alimentés la base OLAP.

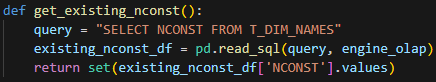
## Migration des données vers la Base OLAP

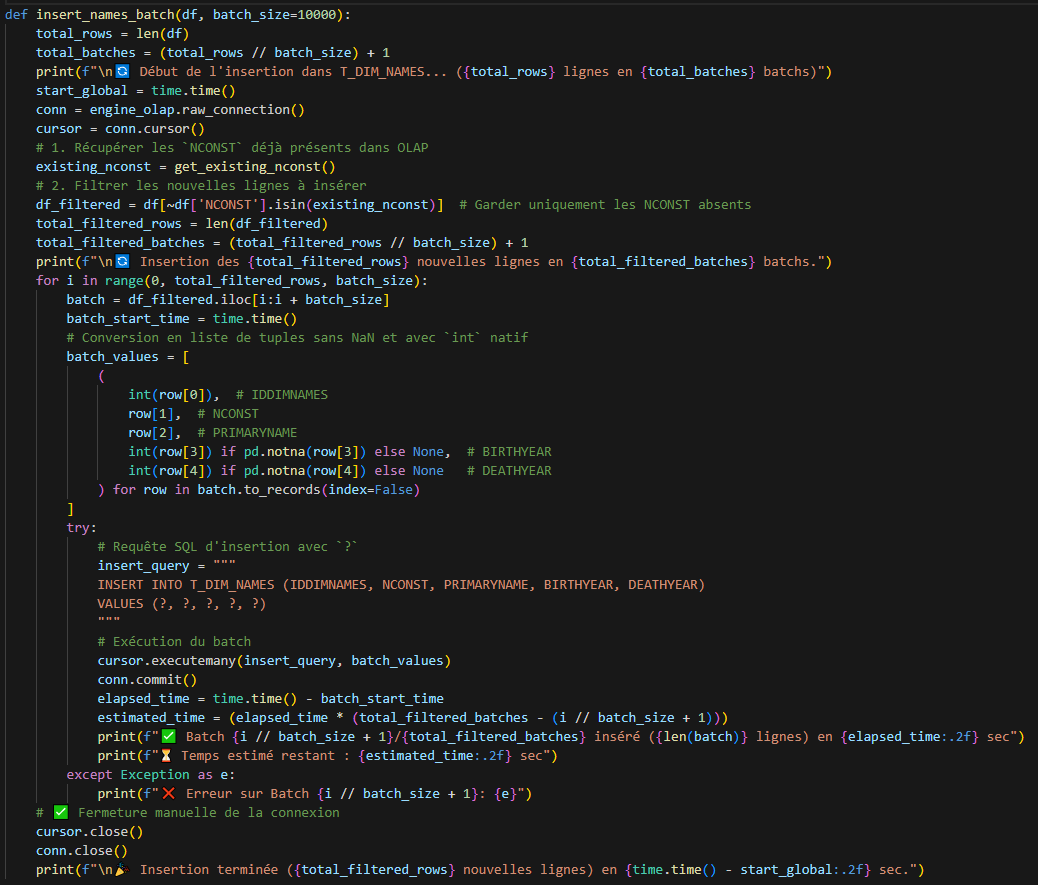
Une seconde transformation, toujours réalisée avec Python, transfère les données vers une base **OLAP** (Online Analytical Processing). La base OLAP est conçue pour les analyses et l'agrégation des données.  
Pour exemple, voici le déroulement de l’insertion de la table T\_DIM\_NAMES :  
Une fois les variables créées on réalise 5 fonctions différentes : -

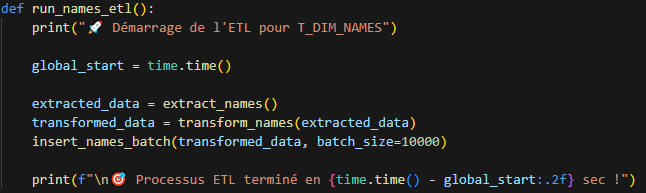
* extract\_names : sert à récupérer les données de la base OLTP.
* Transform\_names : Transforme les données pour qu’elles soient conforme à la dimension.

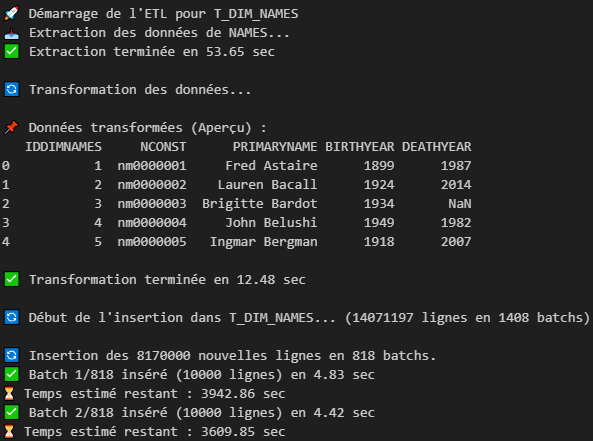


* get\_existing\_nconst : Cette fonction permet d’éviter les doublons d’insertion, car elle vérifie, avant l’insertion si dans la base OLAP que nconst, qui est l’identifiant de la table dans OLTP, n’est pas déjà présent dans la base.



* Insert\_names\_batch : Cette fonction est lancée après les 2 précédentes, elle insère les données dans la dimension en batch.
* run\_names\_etl : Lance l’ETL



Voici le résultat du code :  


Une fois que les dimensions et les tables de faites ont été alimentées, on va pouvoir exploiter la base OLAP via POWER BI, un outil de visualisation permettant de créer des rapports et des tableaux de bord interactifs.

# Rapport BI

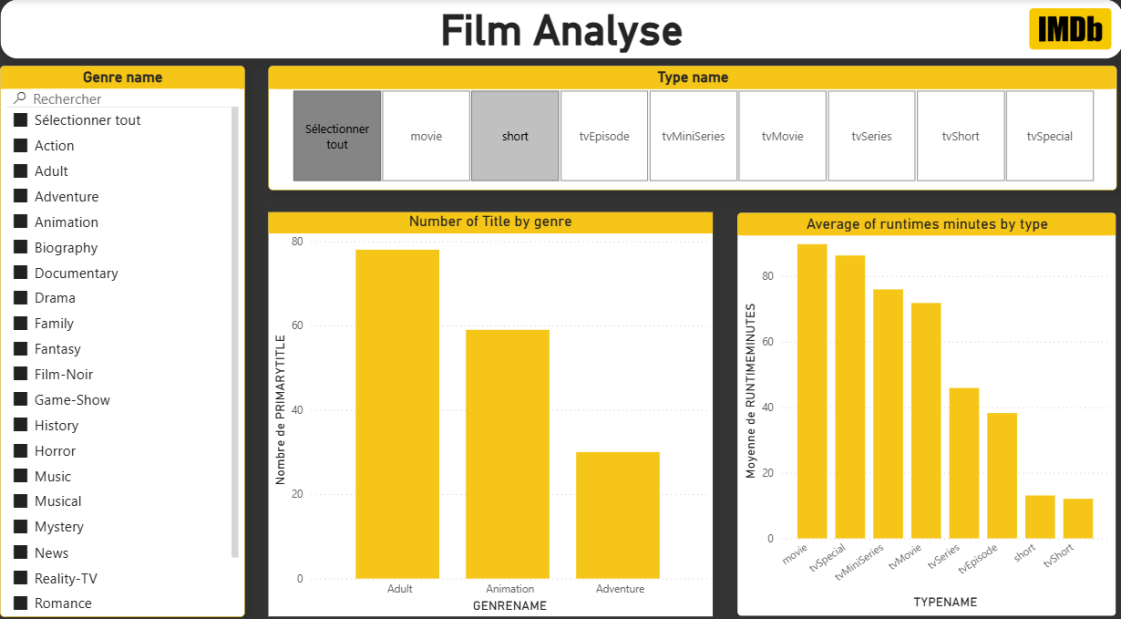
Nous avons choisi de créer deux tables de faits distinctes :

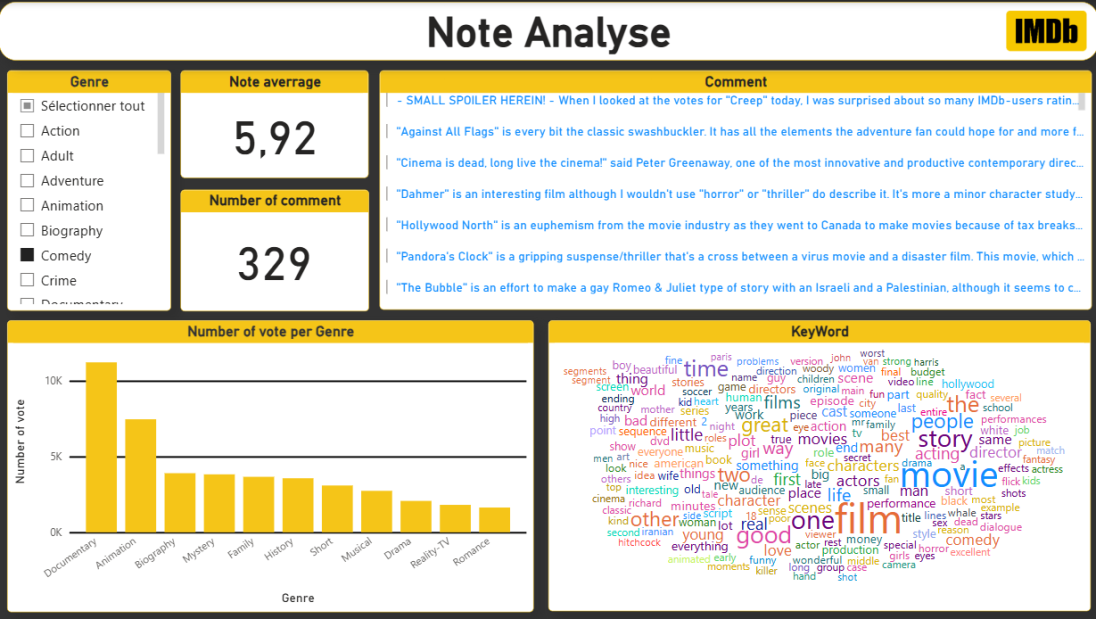
* T\_FACT\_KEYWORD : Cette table permet d’analyser les commentaires en les associant aux films et aux genres correspondants. Elle offre une vision détaillée des avis des spectateurs en fonction des catégories de films.
* T\_FACT\_MEDIA : Cette table permet d’analyser les acteurs en lien avec les films, mais aussi d’étudier les genres cinématographiques et les rôles joués dans ces productions. Elle offre ainsi une approche globale pour comprendre l’impact des acteurs, des genres et des rôles sur les films.

Voici nos pages de rapport réalisé pour analyser les commentaires :  
Cette 1ère page nommé « Film Information » nous donne des informations sur les titres, comme leur genre, leur type (film, série, etc.), la note moyenne des commentaires et le nombre de commentaire.



La 2ème page est centrée sur des analyses plus spécifiques sur les films, on retrouve deux filtres, un filtre sur le genre (action, aventure, drama, etc.) et l’autre sur le type. On a aussi 2 graphes, le premier qui nous permet de montrer le nombre de titre en fonction du genre. Le second quant à lui permet de définir la durée moyenne d’un titre en fonction du type.



La 3ème et dernière page nommée « Note Analyse » est centrée sur une analyse poussée sur les commentaires, on a un filtre qui permet de trier les commentaires par le genre du titre.  
Deux cartes d’information qui donne le nombre de commentaires et la note moyenne de ceux-ci. Un graphe représentant le nombre de vote en fonction du genre du film. Et pour finir un nuage de mot représentant les mots clefs les plus présents dans les commentaires.  


Nous avons trouvé cela pertinent de réaliser un rapport sur les commentaires, et ainsi analyser les mots clefs, car notre modèle IA est « Back of Words » qui est basé sur le même principe.

# Modèles IA produits et démarche de développement et d’optimisation

Notre implémentation des modèles d’IA repose sur une approche de classification des sentiments des critiques de films IMDb et l’attribution d’une note automatique à ces critiques.

Les modèles que nous avons développés sont principalement basés sur Bag of Words (BOW) et Support Vector Machine (SVM), et exploitent des données textuelles pour la classification.

## Données d’apprentissage

Les modèles sont entraînés et  structuré de la manière suivante :

* Dossier pos : Contient des critiques positives (chaque fichier texte inclut une critique et la note de l’utilisateur).
* Dossier neg : Contient des critiques négatives.
* Dossier unsup : Contient des critiques non étiquetées utilisées pour le prétraitement et l’amélioration du modèle.

## Prétraitement des données

Avant d’entraîner les modèles, plusieurs étapes de transformation des données ont été appliquées :

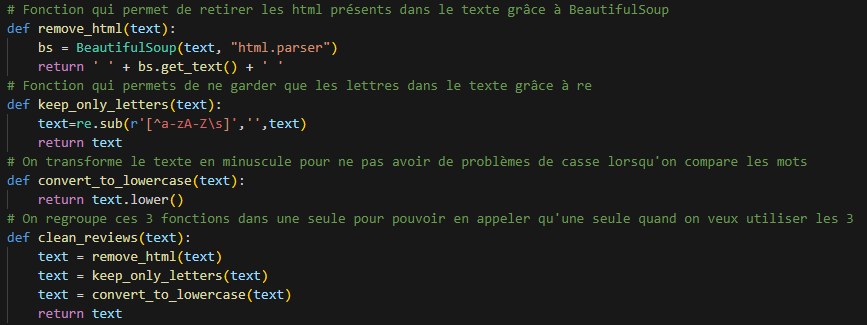
* Suppression des balises HTML à l’aide de BeautifulSoup pour nettoyer le texte.
* Conversion en minuscules pour homogénéiser les critiques.
* Suppression des stopwords (mots courants sans valeur informative comme "the", "is", "and").
* Vectorisation des critiques avec CountVectorizer (modèle Bag of Words).

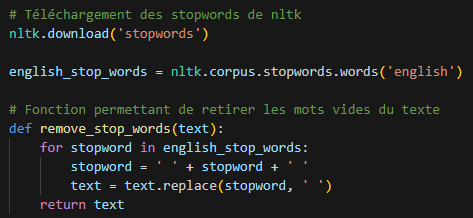
## Types de modèles utilisés

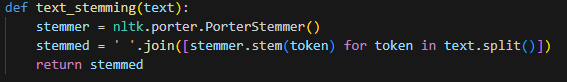
### Modèle 1 : Classification binaire du sentiment

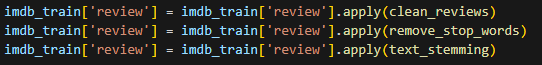
* Objectif : Classifier automatiquement une critique comme positive ou négative.
* Nettoyage des données :

Afin de pouvoir utiliser le Bag Of Words de manière optimisée, il est nécessaire de nettoyer les données utilisées, ici les commentaires. On souhaite donc retirer tous les caractères spéciaux, la casse, les mots qui n’ont pas de sens comme This, In, For… et finalement on applique du text stemming au texte afin de conserver uniquement la partie ayant du sens dans le mot, ce qui permet aussi de regrouper les mots ayant le même sens mais une terminaison différente comme les noms, adjectifs et verbes pour le même mot. Pour ceci, le code suivant a été réalisé :

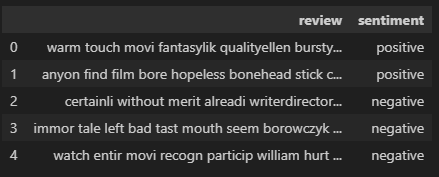








Exemple de commentaires nettoyés :



* Approche : Modèle Support Vector Machine (SVM) avec une représentation Bag of Words (BOW).

Bag Of Words : Représentation vectorielle d’un texte avec le nombre d’apparition de chaque mot.  
Pour créer celui-ci, on utilise un CountVectorizer :



Les paramètres utilisés sont binary=False pour avoi le nombre d’apparition du mot et non seulement savoir s’il apparait, ngram\_range=(1,2) permet d’utiliser des unigrammes et bigrammes, c’est-à-dire un mot ou un groupe de 2 mots. Finalement, min\_df=0.01 permet de conserver uniquement les mots qui apparaissent dans 1% des commentaires.

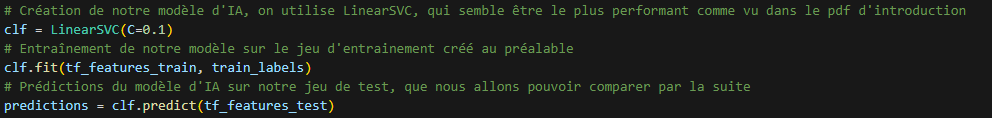
Pour obtenir ces paramètres, des test ont été effectués, min\_df entre 0.001 et 0.05, les trigrammes ont aussi été essayés, mais les meilleurs résultats pour un temps d’exécution assez rapide ont été obtenus avec les paramètres que l’on utilise ici.

Ensuite, on applique ce vecteur à nos données :



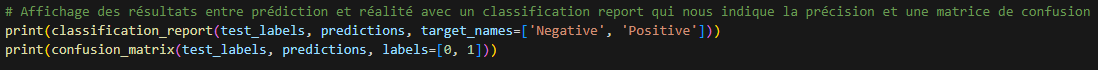
Création du modèle LinearSVC : La fonction LinearSVC de scikit-learn est une implémentation de **SVM (Support Vector Machine)** linéaire, utilisée pour résoudre des problèmes de classification. Contrairement à SVC qui permet de créer des modèles SVM avec noyau (kernel), LinearSVC est spécifiquement destiné aux cas où la séparation entre les classes est linéaire, c'est-à-dire lorsque les données peuvent être séparées par une droite ou un hyperplan.

Le code utilisé est donc le suivant :

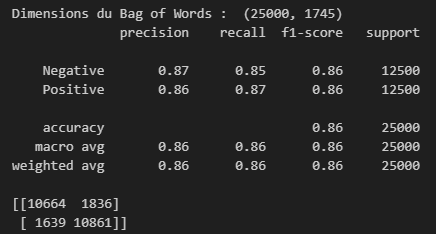


On ajoute le paramètre C=0.1 afin de faciliter la convergence du modèle.

On utilise ensuite un fit sur notre jeu d’entraînement et finalement le predict afin d’avoir les résultats de notre modèle entraîné.  
Pour afficher ces résultats, on utilise la matrice de confusion et le rapport de classification ou l’on peut lire la précision :



Voici les résultats obtenus :



On trouve une précision de 86% pour la prédiction du sentiment.

Tout ce code est disponible dans le fichier BOWmodele1.ipynb.

### Modèle 2 : Attribution automatique d’une note (1 à 10)

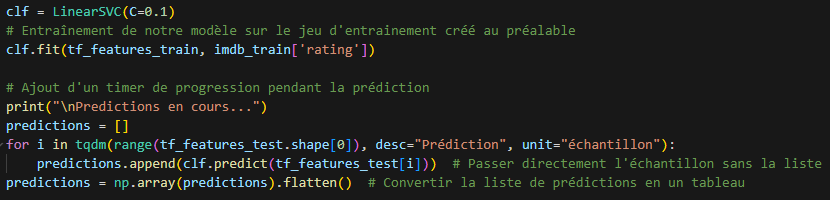
* Objectif : Prédire la note sur 10 d’une critique.

Nettoyage des données identique à la partie précédente.

* Approche : Modèle de régression linéaire appliqué aux données vectorisées.

Le Bag Of Words est crée de la même manière et avec les mêmes paramètres que dans le modèle précédent.

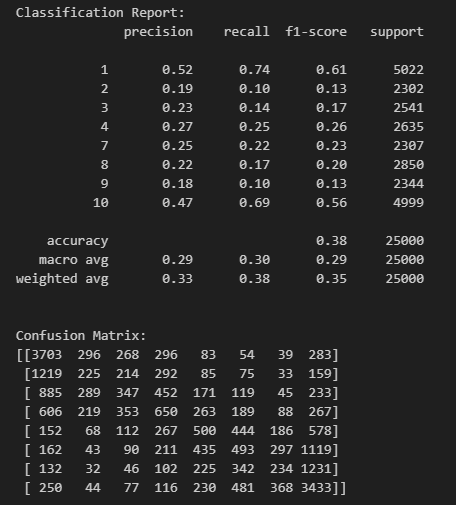
Création du modèle avec LineaSVC :



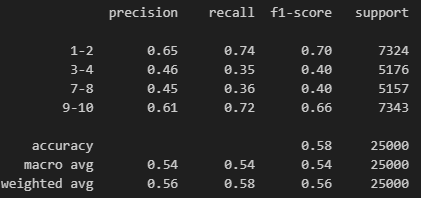
Même principe que précédemment cependant, comme il y a maintenant 8 sorties au lieu de 2, on rajoute une jauge tqdm qui permet de suivre l’avancement du programme :

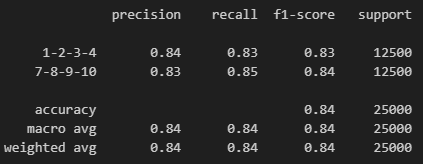


Les résultats obtenus sont alors les suivants :

  
  
On remarque alors tout de suite que la précision est descendue à 38%, de plus, elle est beaucoup plus élevée pour la note de 1 et la note de 10, en effet, le modèle est biaisé vers les extrêmes car ils sont plus faciles à reconnaître.

Nous avons donc décidé de regrouper les notes par 2 puis par 4 afin de quand même vérifier la précision dans ces cas-là. Voici les résultats obtenus :





On retrouve alors ici une précision de 58%, qui est plutôt correcte et une précision de 84% pour l’équivalent du sentiment ce qui est très proche des résultats précédents.

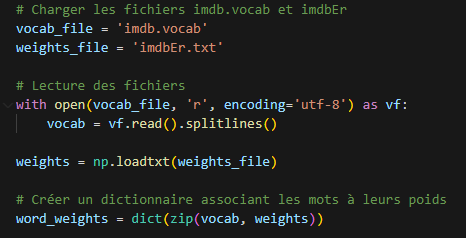
Tout ce code est disponible dans le fichier BOWmodele2.ipynb.

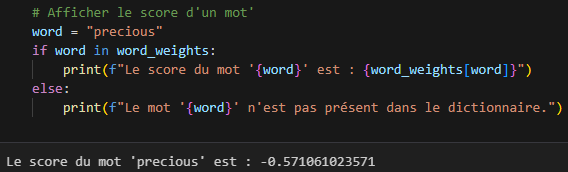
### Modèle 3 : Score gradué du sentiment

* Objectif : Sortir d’une classification stricte binaire (positif/négatif) et proposer une gradation du sentiment (Très Négatif, Négatif, Neutre, Positif, Très Positif).
* 1e solution :

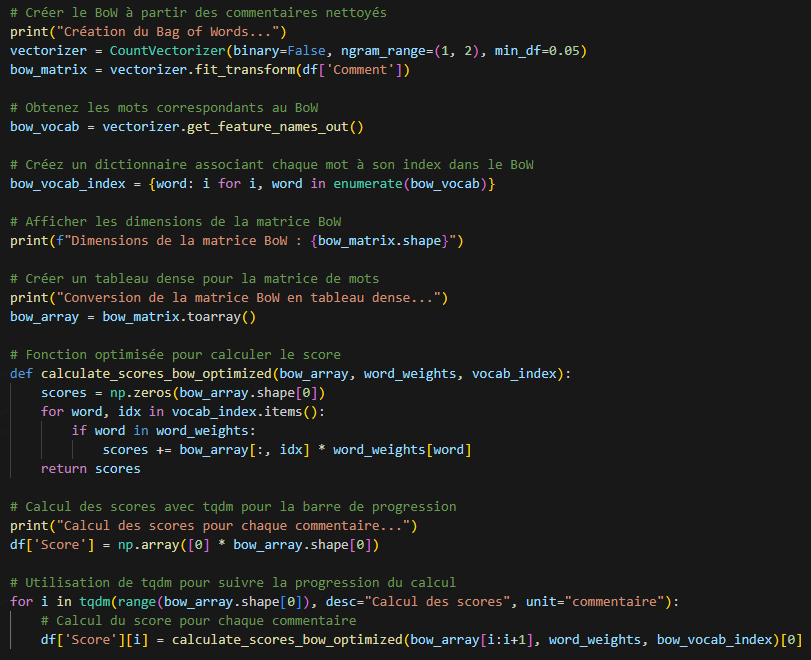
Ici, un nouveau système de gradation est créé, il n’est donc pas possible d’entraîner un modèle comme précédemment.

Il faut donc trouver une solution permettant d’associer ce grade au commentaire. La première solution envisagée est l’utilisation du dictionnaire imdb.Er et du vocabulaire imdb.voc fournis avec les données. On peut alors attribuer un score à chaque mot du commentaire, calculer un score total et affecté un grade en fonction de cette note. Voici le code utilisé pour réaliser ceci.

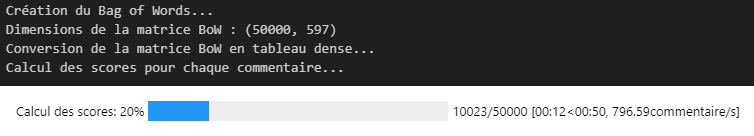


Exemple de score attribué à un mot : 

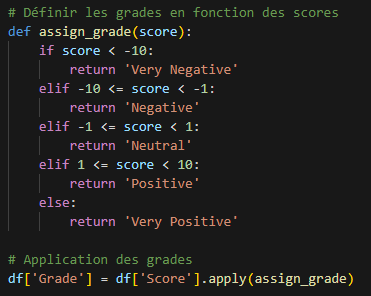
On utilise encore une fois un Bag Of Words afin de garder les mots les plus significatifs et accélérer le calcul des grades afin que le programme ne soit pas trop long et gourmand.



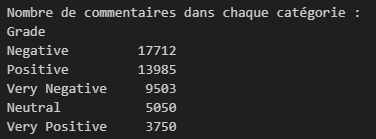
Une jauge tqdm est aussi ajoutée afin de suivre l’avancée du calcul des notes.



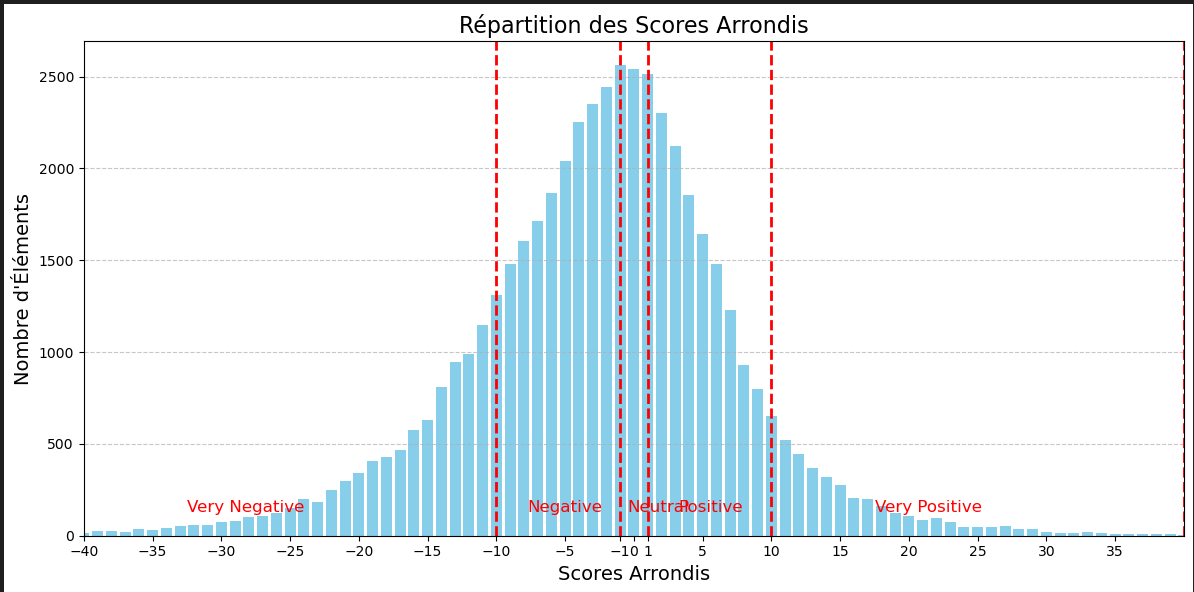
Finalement, on associe un grade en fonction du score qui vient d’être calculé :



On obtient alors :

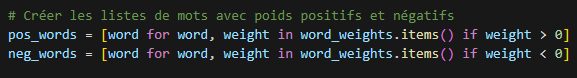


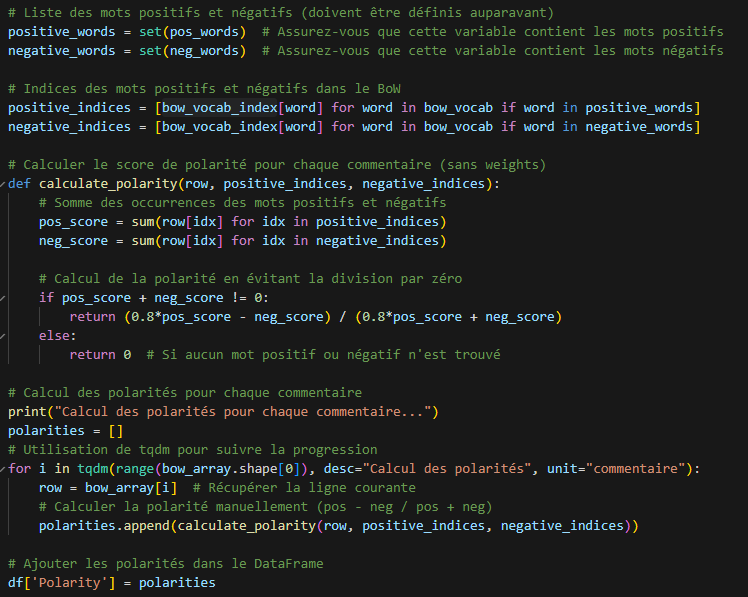
Ici, on peut observer que le score est très souvent négatif avec un total de 27 000 commentaires négatifs contre 17 000 commentaires positifs, alors que nous savons que les données contiennent autant de commentaires positifs que négatifs. On le voit particulièrement en traçant le graphique des scores :



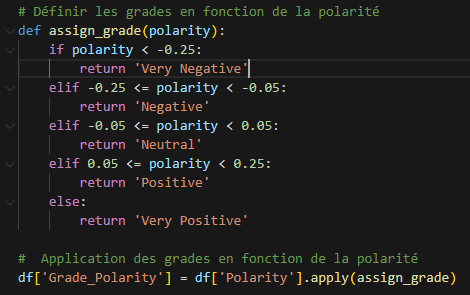
Cette solution ne semble alors pas la plus adaptée à la situation, nous avons donc alors chercher de nouveaux vocabulaires comme Harvard IV ou VADER, mais les résultats ne semblaient pas satisfaisants non plus (tests disponibles dans le fichier BOWmodele3dictionnaire2). Cependant, ceux-ci introduisaient la polarité, qui est calculée de la sorte :

P = (MP-MN)/(MP+MN) avec MP le nombre de mots positifs et MN le nombre de mots négatifs.  
Ce qui permet alors d’avoir un résultat entre -1 et 1 qui permet d’associer un grade. Cette polarité a donc été utilisée comme solution avec le dictionnaire utilisé :



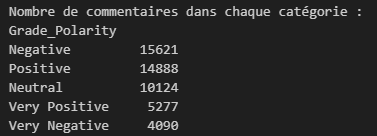


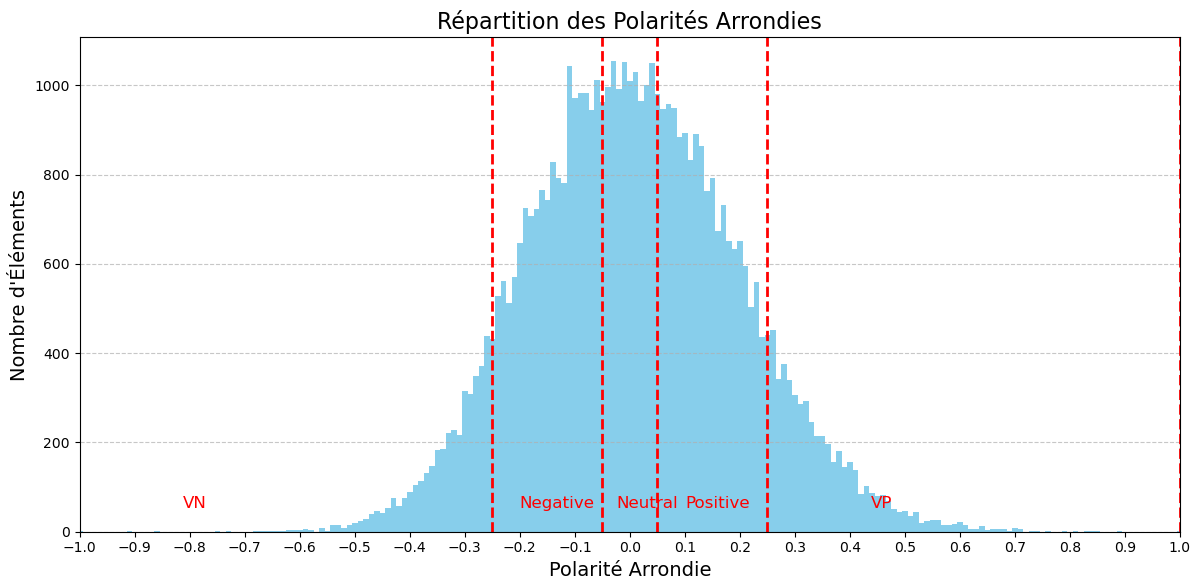
Ici, un facteur a été rajouté devant le score positif car le dictionnaire contient 2 fois plus de mots positifs que de mots négatifs ce qui biaise les résultats qui se retrouvent alors majoritairement positifs.



Attribution du grade en fonction de la polarité

On obtient alors la répartition suivante :



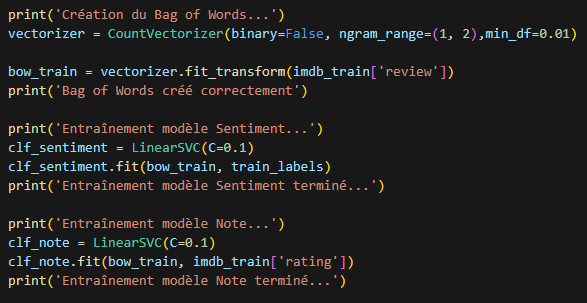


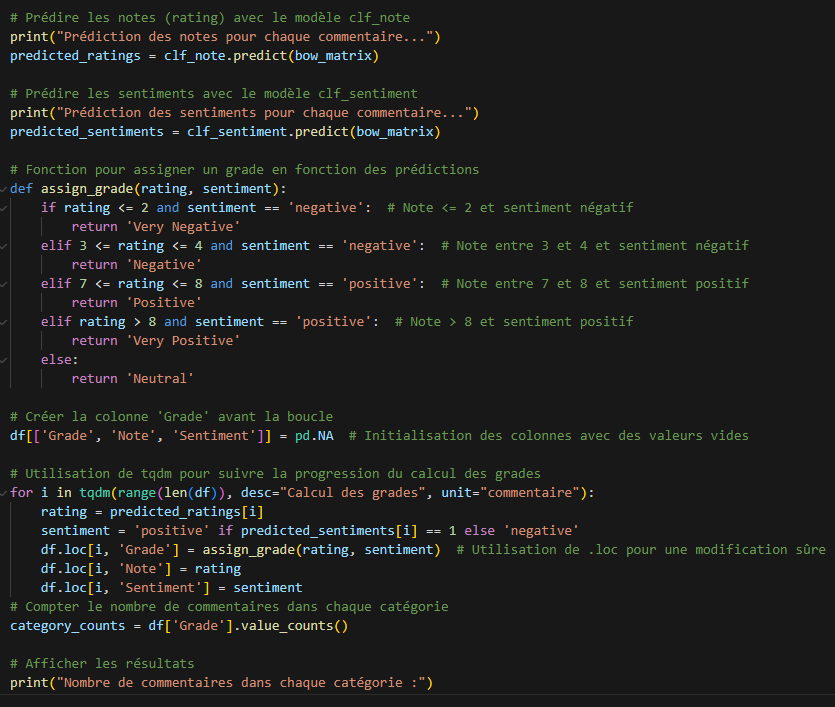
On retrouve alors une répartition qui semble très correcte et correspond au 50% de positif et négatif. Ce code est disponible dans le fichier (BOWmodele3.ipynb).

Cependant en effectuant des tests et en regardant certains commentaires existants, les résultats ne semblaient pas très satisfaisants. Nous avons donc essayé une 2ème solution.

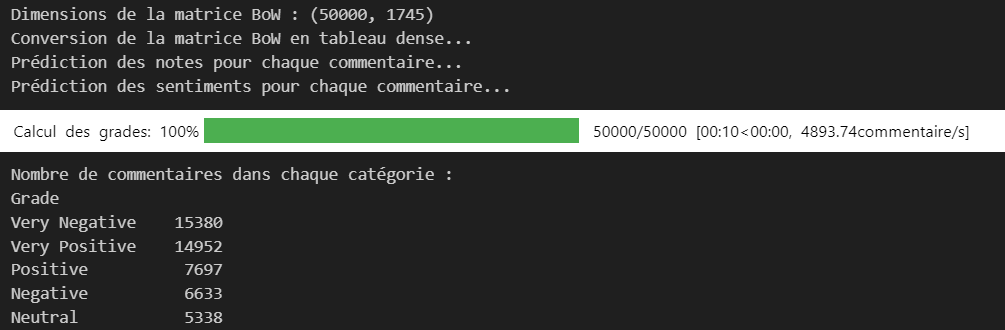
* 2ème solution :

La 2ème solution envisagée a alors été de réutiliser les modèles auparavant entraînés pour le sentiment et la note. On utiliserait ces deux modèles sur les commentaires puis en combinant leurs résultats, il est possible d’associe un grade. Le code permettant de réaliser ceci est le suivant :





Les résultats obtenus sont alors les suivants :



On retrouve alors une répartition avec 22 000 commentaires positifs et négatifs, ce qui semble être une très bonne répartition, on remarque aussi une grande quantité de Très Positifs et Très Négatifs qui s’explique par le modèle de Notes qui est biaisé envers les extrêmes.

Finalement, nous avons réalisé un code permettant de connaître le sentiment, la note et le grade attribué par nos 2 modèles en entrant un commentaire IMDB :

