



**Master 1 Économétrie - Statistiques  
Langage de Programmation 1  
2023-2024**

## **ANALYSE DE DONNÉES SPOTIFY**



COLIN Léo / HOARAU Léa / SOPGUOMBUE Brice / VO Nguyen Thao Nhi

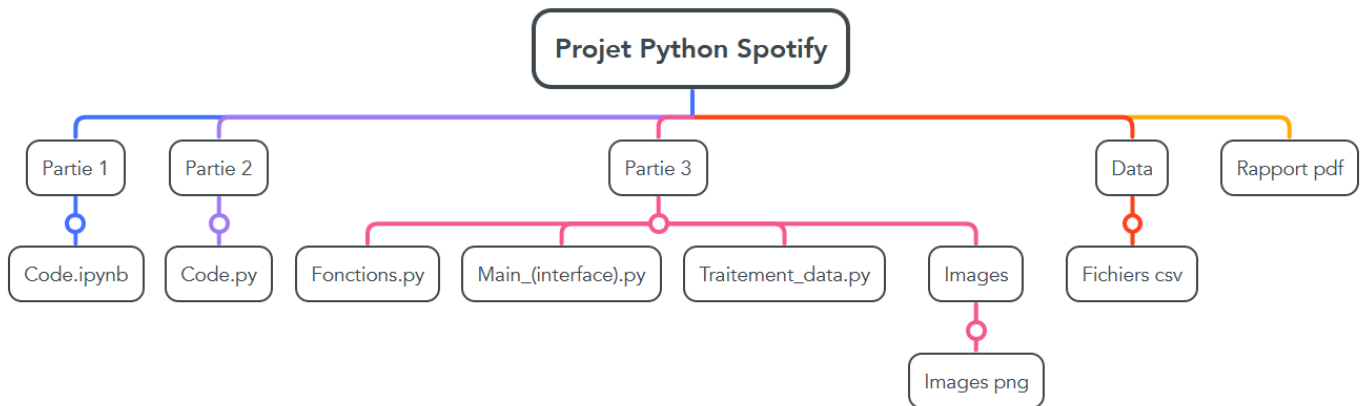
# TABLE DES MATIÈRES

<b>INTRODUCTION</b>	<b>2</b>
<b><u>PARTIE 1:</u> ANALYSE DESCRIPTIVE DES BASES ET VISUALISATION</b>	<b>4</b>
<b><u>PARTIE 2:</u> RECHERCHE DE CONTENU</b>	<b>9</b>
<b><u>PARTIE 3:</u> INTERFACE GRAPHIQUE</b>	<b>11</b>
<b>COMMENT EXÉCUTER LE PROGRAMME ?</b>	<b>16</b>
<b>Nos difficultés...</b>	<b>17</b>
<b>Ce que le projet nous a appris...</b>	<b>19</b>

# INTRODUCTION

Pour ce projet final, le sujet qui nous a le plus inspiré est celui portant sur Spotify. Nous avons utilisé les données provenant de Kaggle<sup>1</sup> que nous analyserons en détail dans la première partie. En ce qui concerne les données Top200, nous n'aurons besoin que du fichier *spotify\_top200\_global* et non ceux des pays séparément.

Notre projet se décompose en un dossier pour chacune des trois parties à traiter dans le cadre de ce projet, un dossier contenant les données et le rapport que vous êtes en train de lire



Nous avons tous travaillé sur la partie 1 qui était une étape descriptive mais primordiale afin de comprendre la suite du projet. Ensuite, deux d'entre nous ont construit les programmes de la partie 2, et les deux autres personnes du groupe ont travaillé l'interface. Cependant, les deuxième et troisième parties (citées ci-dessous) étant intimement liées, il nous a fallu à tous

<sup>1</sup> Source 1 : <https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks>

Source 2 : [Top 200 most streamed songs on spotify 2020 \(kaggle.com\)](https://www.kaggle.com/datasets/yamaerenay/top-200-most-streamed-songs-on-spotify-2020)

prendre connaissance du travail des uns et des autres afin de s'entraider et de comprendre les éventuels problèmes et corrections à effectuer.

Dans la suite de ce rapport, nous allons d'abord présenter une analyse descriptive des bases que nous avons réalisée dans un notebook. Puis nous détaillerons la création des programmes permettant à un utilisateur de faire des recherches selon plusieurs critères. Enfin, nous présenterons l'interface graphique à travers laquelle l'utilisateur pourra effectuer les mêmes recherches mais via à une interface plus intuitive.

# **PARTIE 1: ANALYSE DESCRIPTIVE DES BASES**

## **ET VISUALISATION**

*Au sein du dossier 1 vous trouverez un notebook contenant le code que nous avons réalisé pour la partie 1 du projet.*

Pour commencer, nous effectuons une description classique des données des trois dataframes disponibles : *artists*, *tracks* et *top200*.

Le dataframe *artists* contient des informations sur les artistes. Dans ce dataframe, il y a 1 162 095 observations et 5 variables. Nous y retrouvons l'ID de l'artiste (*id*), le nom d'artiste (*name*), le nombre d'abonnés (*followers*), les genres associés (*genres*) et le niveau de popularité (*popularity*). Les variables *id*, *genres* et *name* sont de type object, tandis que *followers* est de type float et *popularity* est de type integer.

Le dataframe *tracks* possède des informations sur des chansons présentes sur Spotify, datant de 1921 à 2020. Il compte 586 672 observations et 20 variables. Ces informations comprennent, par exemple, le nom de la chanson (*name*), les artistes qui figurent sur la chanson (*artists*), la date de sortie (*release\_date*), le niveau de popularité (*popularity*), ainsi que les caractéristiques de la chanson (*danceability*, *loudness*, *tempo*, etc.). La plupart des variables sont de type float ou integer, à l'exception de *id*, *name*, *artists*, *id\_artists*, et *release\_date* qui sont de type object.

Le dataframe *top200* contient les titres qui font partie du top 200 mondial de l'année 2020. Il compte 73 200 observations avec 6 variables, incluant le nom de l'artiste (*artist*), la date (*date*), le rang de la chanson dans le classement (*rank*), le nombre d'écoutes (*streams*) et le nom de la chanson (*title*), et le pays (*country*). Ici, les variables *artist*, *country*, *date* et *title* sont de type object, tandis que *rank* et *streams* sont de type integer.

Nous allons maintenant répondre aux questions de la partie 1.

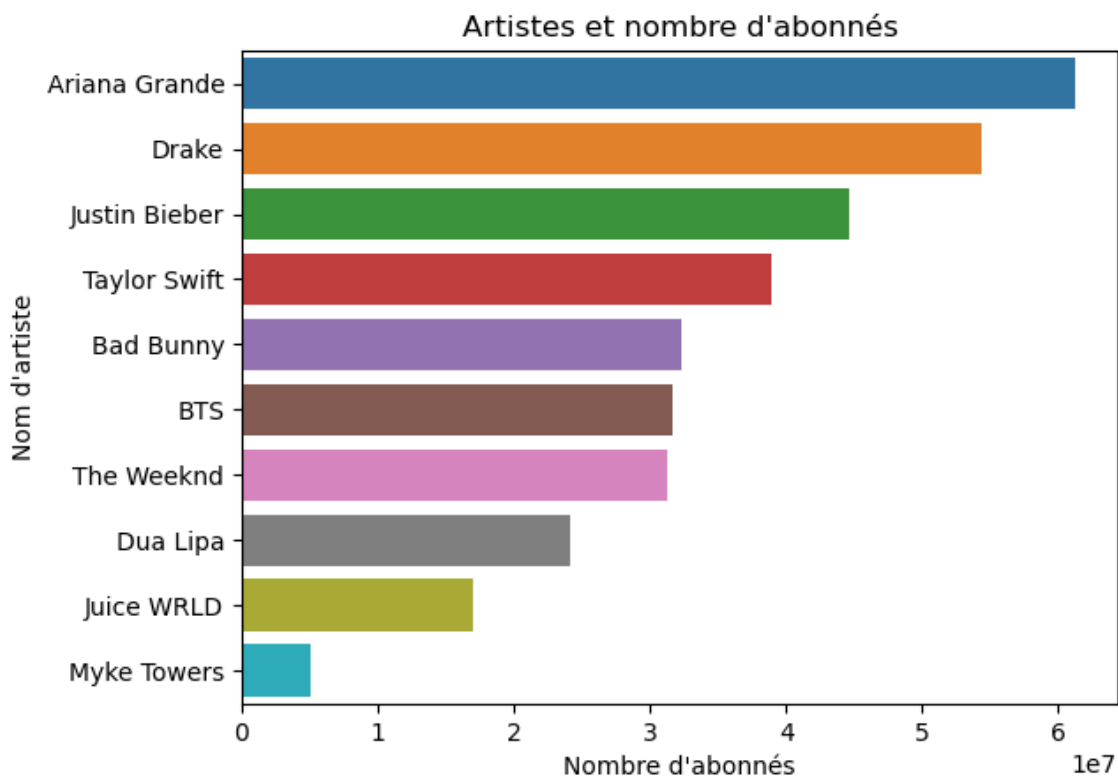
**Première question:** Quels sont les 10 artistes les plus populaires ? Afficher graphiquement leur nombre d'abonnés par ordre décroissant.

Nous utilisons le dataframe *artists* pour obtenir des informations sur le nom d'artiste, son niveau de popularité et son nombre d'abonnés. Pour savoir qui sont les 10 artistes les plus populaires, il existe deux façons de procéder : soit trier le dataframe par popularité décroissante, soit par nombre d'abonnés décroissant, puis afficher les 10 premiers résultats.

Si le dataframe est trié par popularité décroissante, nous constatons que Justin Bieber est l'artiste le plus populaire avec le niveau de popularité égale à 100. En revanche, si le dataframe est trié par le nombre d'abonnés, c'est Ariana Grande qui est la plus populaire avec 61 301 006 abonnés. Pour la présentation graphique, étant donné qu'il est demandé d'afficher leur nombre d'abonnés par ordre décroissant, nous choisissons de trier le dataframe par nombre d'abonnés décroissant.

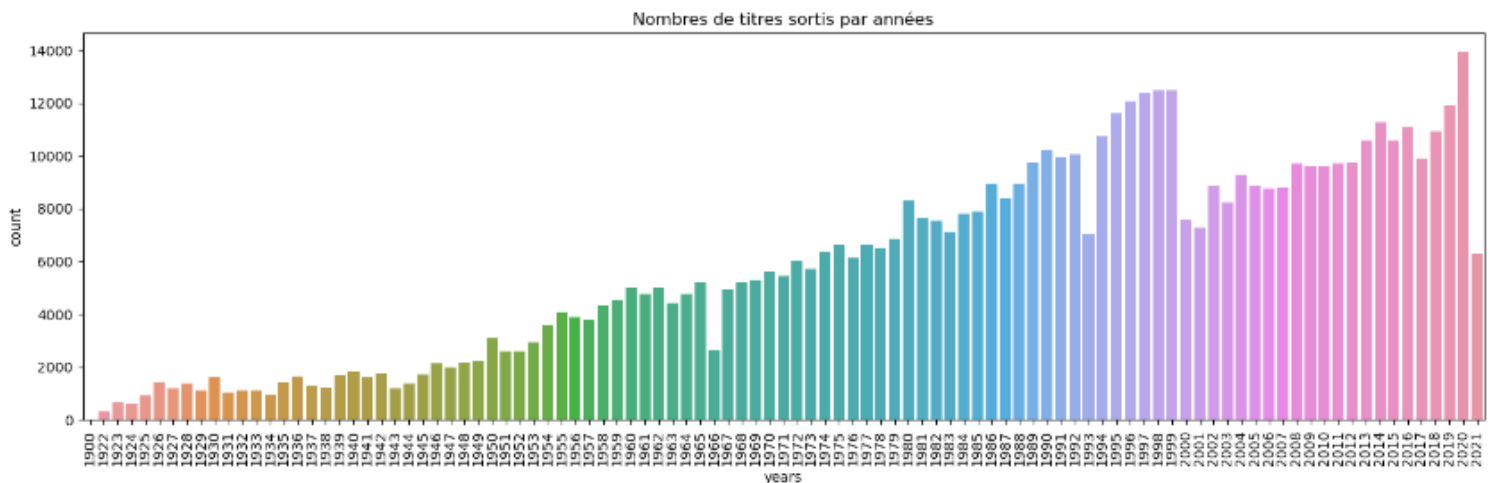
Les 10 artistes les plus populaires sont donc Ariana Grande, Drake, Justin Bieber, Taylor Swift, Bad Bunny, BTS, The Weeknd, Dua Lipa, Juice WRLD et Myke Towers.

Le graphique ci-dessous présente les 10 artistes les plus populaires en fonction de leur nombre d'abonnés, utilisant les données du dataframe *artists*.



**Deuxième question:** Calculer le nombre de chansons sorties chaque année. Représenter graphiquement les résultats.

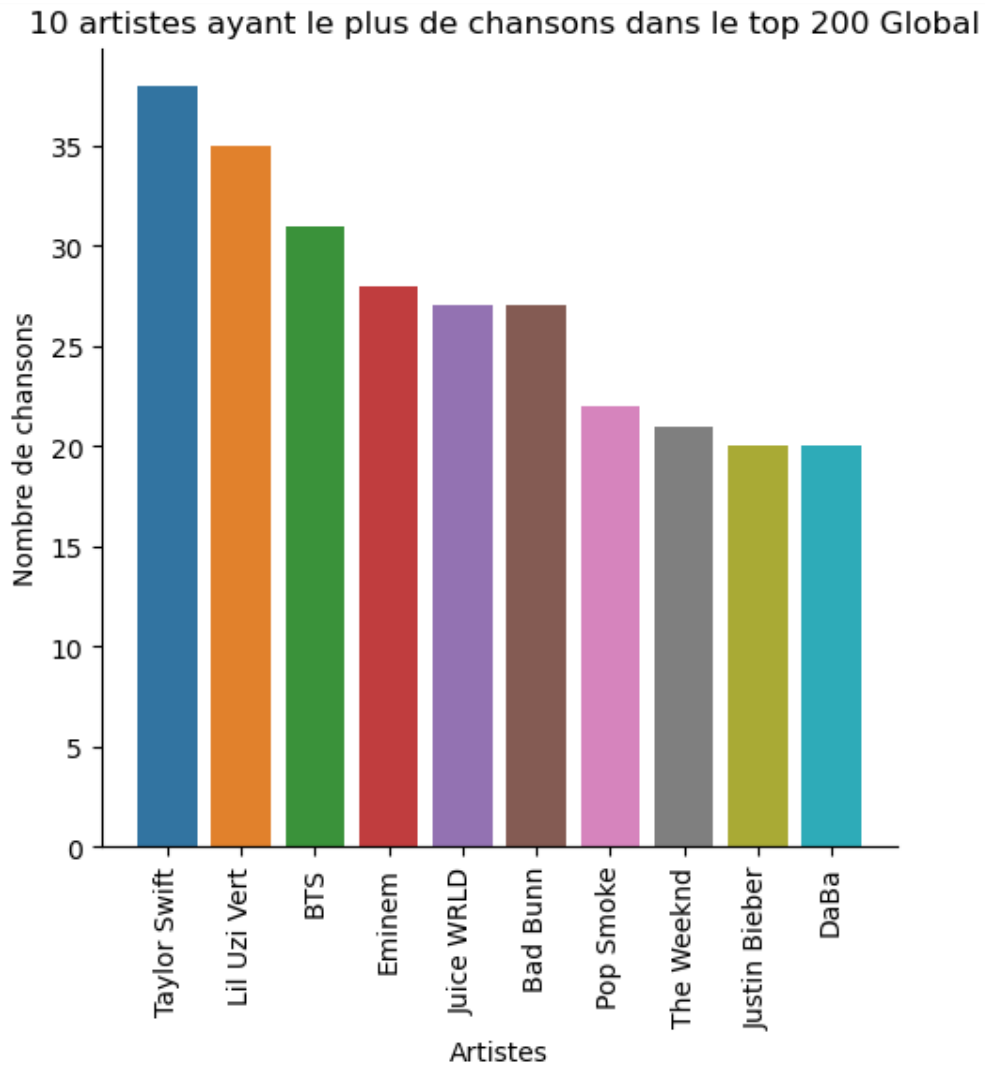
Le dataframe *tracks* est utilisé ici afin de manipuler les chansons et leur date de sortie. Pour travailler avec les dates, il est nécessaire de transformer la variable *release\_date* en type *DateTime* et de changer son format de "%Y-%m-%d" à "%Y". Ensuite, nous comptons le nombre de titres distincts sortis chaque année et les trions par année croissante.



Ce graphique représente le nombre de chansons sorties pour chaque année, de 1921 à 2021. Nous constatons donc une forte augmentation du nombre de titres sortis chaque années entre ces deux dates.

**Troisième question:** Quels artistes ont le plus de chansons distinctes dans le top 200 Global ? En cas d'égalité, les ordonner par nombre de streams cumulés décroissants. Représenter graphiquement les résultats.

Nous utilisons le dataframe *top200* afin de recueillir des informations sur les chansons présentes dans le top 200 Global de 2020, les artistes qui y figurent ainsi que leur nombre de streams associé. Nous comptons le nombre de titres distincts et le nombre total de streams pour chaque artiste, puis nous les trions par nombre de titres de façon décroissante. En cas d'égalité, nous les trions par le nombre de streams cumulés de chaque artiste. Pour une meilleure visualisation, il n'affiche que les 10 artistes ayant le plus grand nombre de chansons dans le top 200 Global.



Les 3 artistes ayant le plus de chansons dans le top 200 Global sont Taylor Swift avec 38 chansons, Lil Uzi Vert avec 35 chansons et BTS avec 31 chansons.

**Quatrième question:** Existe-t-il un lien entre la popularité d'une chanson et les autres critères présents dans les données?

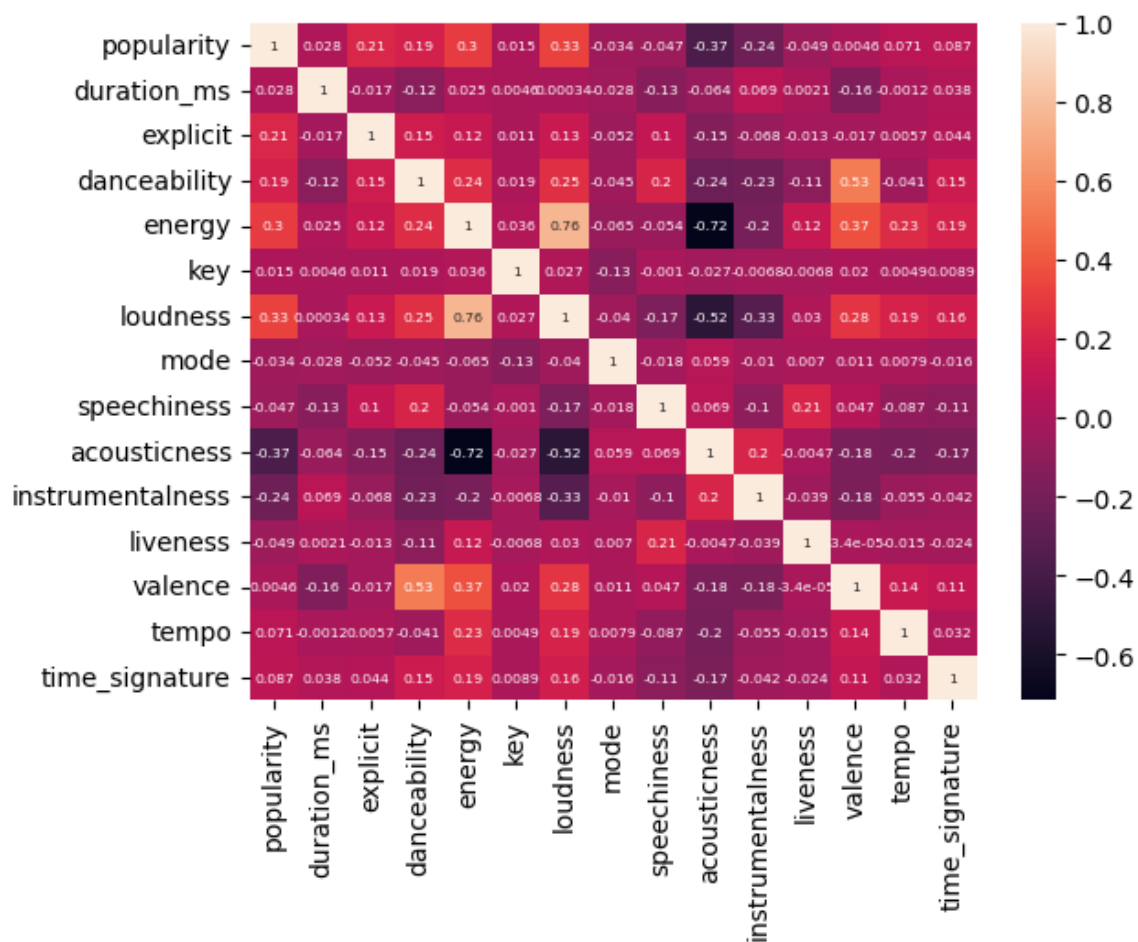
En se basant sur le dataframe *tracks*, le graphique ci-dessous montre que, pour une chanson:

- Les coefficients de corrélation entre la popularité et deux critères, l'énergie et le volume global, sont de 0.3 et 0.33 respectivement. Cela signifie qu'il y a une corrélation modérée et positive entre la popularité et ces deux variables.



- D'autre part, le coefficient de corrélation entre la popularité et l' "acousticness" est égal à -0.37 suggérant qu'il existe un lien négatif et modéré entre la popularité et l' "acousticness".
- Concernant les autres critères, les coefficients de corrélation avec la popularité sont proches de 0, indiquant l'absence d'un lien significatif entre la popularité d'une chanson et ces critères.

En conclusion, on peut affirmer qu'il n'y a pas de lien fort entre la popularité d'une chanson et les critères présents.



Nous allons passer à la création de fonctions pour effectuer des recherches.

## **PARTIE 2 : RECHERCHE DE CONTENU**

*Au sein du dossier 2, vous trouverez un fichier (.py) contenant le code que nous avons réalisé pour la partie 2 du projet.*

Dans cette partie, il faut construire un programme qui permet de retourner des résultats en fonction soit du nom d'artiste, soit du titre de chanson, soit de l'année de sortie et du genre de musique donnés par l'utilisateur.

Avant de réaliser les fonctions de recherche de contenu, il nous faut tout d'abord nettoyer les données une fois qu'elles sont importées. Nous utilisons Pandas pour travailler avec des dataframes, RegEx pour enlever les caractères spéciaux, non-alphanumérique en gardant toujours les espaces dans les dataframes et Datetime pour manipuler les dates.

De plus, afin d'éviter des problèmes liés à la casse des caractères (minuscules et majuscules) dans les informations saisies, nous mettons toutes les colonnes dans les dataframes contenant des noms d'artistes, des titres de chansons et des genres de musique en minuscule avec la fonction *str.lower()*.

Nous remarquons qu'il y a des doublons dans la colonne *name* du dataframe *artists*. Pour résoudre ce problème, pour chaque nom, nous ne gardons que la ligne dont le nombre d'abonnés est le plus élevé, parce que la plupart des doublons ont un nombre d'abonnés égal à 0. Un nouveau dataframe *artists\_unique* est créé.

À partir du dataframe *top200*, nous comptons, pour chaque artiste, le nombre de ses chansons dans le top 200 global de 2020 afin d'utiliser dans la fonction de recherche. Un dataframe *top\_artists* est créé.

La première fonction *Artiste\_info()* demande à l'utilisateur de saisir un nom d'artiste et retourne son nombre d'abonnés, les trois chansons les plus populaires, les trois chansons les plus récentes et le nombre de chansons qu'il a dans le top 200 global de 2020. Ici, l'information saisie est

également mise en minuscules. Cette fonction vérifie d'abord si le nom d'artiste donné existe bien dans le dataframe *artists\_unique* et *tracks* ou non. Sinon, elle retourne un message d'erreur. Si oui, elle continue à exécuter.

- Pour le nombre d'abonnés la fonction renvoie le nombre d'abonnés correspondant à cet artiste dans le dataframe *artists\_unique*.
- Pour les trois chansons les plus populaires, à partir du dataframe *tracks*, nous filtrons un sous-dataframe pour cet artiste qui contient ses chansons et le niveau de popularité de chaque chanson, puis nous trions ces chansons par popularité décroissante. La fonction affiche les trois premiers résultats qui sont les trois chansons les plus populaires.
- Pour les trois chansons récentes, à partir du dataframe *tracks*, nous filtrons un sous-dataframe pour cet artiste qui contient ses chansons et la date de sortie de la chanson, puis nous trions les chansons par date de sortie décroissante. La fonction affiche les trois premiers résultats qui sont les trois chansons les plus récentes.
- Ensuite, la fonction vérifie si l'artiste a au moins une chanson dans le top 200 global de 2020. Elle cherche si ce nom existe dans le dataframe *top200*. Sinon, elle retourne un message d'erreur. Si oui, elle renvoie le nombre de chansons correspondant à cet artiste dans le dataframe *top\_artists*.

Dans le deuxième point de la partie 2, il nous était demandé de créer un programme qui permet de rechercher un titre de chanson. Le résultat devait retourner toutes les chansons qui y correspondent, et n'afficher que les 20 premières s'il y en avait plus que cela. Afin de créer ce programme, nous avons dû utiliser le dataframe *tracks*.

- Il a d'abord fallu traiter le problème de la sensibilité à la casse dans notre fonction *song()*, en effet, le but était de faire en sorte que si l'utilisateur entre un titre de chanson en utilisant majuscule, minuscule ou les deux à la fois cela n'ait pas d'impact sur le résultat. Pour cela, nous avons utilisé *.lower()* à la fois sur ce que saisit l'utilisateur mais aussi sur la colonne qui contient les titres de chansons à l'intérieur du data frame.
- Pour aller chercher les données dont nous avons besoin, il a fallu créer une variable *recherche* qui va prendre les données qui nous intéressent, telles que le nom de l'artiste,

le titre de la chanson et la popularité, dans *tracks*, selon ce qu'a saisi l'utilisateur (soit *titre*). Si le titre de chanson n'est pas reconnu ou qu'il n'existe pas dans le dataframe, la fonction va en informer l'utilisateur, sinon, toutes les chansons qui correspondent au texte saisi seront affichées par ordre de leur popularité décroissante.

- Cependant, il y avait parfois le même résultat de chanson affiché plusieurs fois, c'est pourquoi nous avons dû utiliser *groupby().agg()* qui permet de grouper les données par nom de chanson et d'artiste et de ne garder que la chanson qui est la plus populaire. Le résultat affiche les 20 chansons les plus populaires.

Dans la dernière sous-partie de la partie 2, nous avons créé une fonction *Angenre()* qui renverra toutes les chansons qui correspondent à deux critères simultanément : l'année de sortie et le genre de musique. Le résultat devra afficher les chansons ordonnées par popularité d'artiste décroissante, puis s'il y a plusieurs chansons pour un même artiste, elles devront également être ordonnées par ordre de leur popularité décroissante.

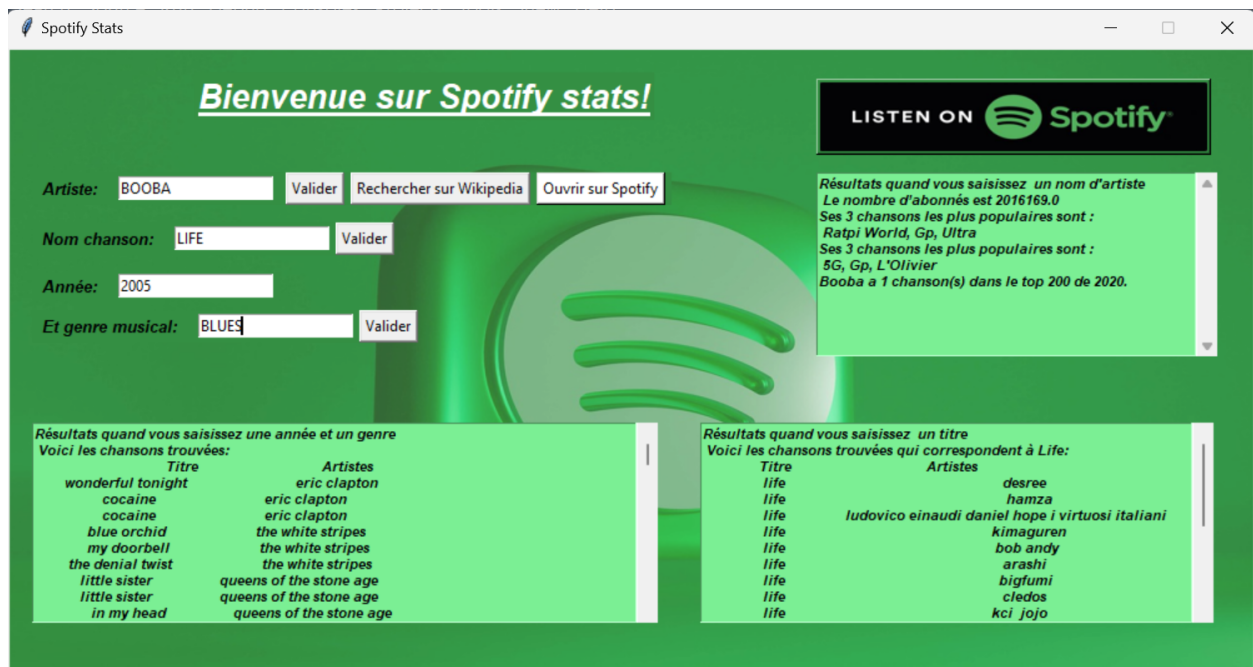
- Nous avons dû rassembler les deux dataframes pour que la fonction puisse faire les tris par popularité d'artiste et de chanson à la fois.
- Nous avons toujours utilisé les mêmes propriétés que les fonctions précédentes, comme la mise en minuscule du genre saisi.
- De plus, afin que l'année ne pose pas de problème, nous avons créé une nouvelle colonne *release\_year* qui renvoie l'année uniquement, ainsi le programme comprendra donc bien l'année sous format YYYY.
- Si l'année ou le genre saisis ne sont pas des valeurs que contient la fusion, l'utilisateur en sera informé. Enfin, si les deux critères sont valides et contenus dans la fusion, la fonction devrait bien afficher les chansons de la bonne année et du bon genre en triant par popularité d'artiste et de chanson décroissante, soit de la plus connue à la moins connue.

Nous allons maintenant passer à la création de l'interface graphique.

## PARTIE 3 : INTERFACE GRAPHIQUE

Le dossier 3 se décompose en 3 fichiers PY (“Main”, “Fonctions” et “traitement\_data”) et un sous-dossier contenant deux images. Le script “Main” est le code permettant de faire fonctionner l’interface. Ce script fait appel aux images présentes dans le sous-dossier “Images” et aux fonctions du module “Fonctions”. Le module “Fonctions” appelle lui-même des objets du script “traitement\_data”. Ce dernier importe les données présentes dans le dossier “Data” avant d’en faire un traitement préliminaire.

Aperçu de notre interface graphique:



Afin de faire fonctionner correctement notre interface graphique, nous avons dû modifier nos fonctions de la partie 2. Ces nouvelles fonctions ont toujours la même finalité mais se comportent différemment. Contrairement, aux fonctions de la partie 2, certaines fonctions de la partie 3 prennent des arguments en entrée (seule façon que nous avons trouvé pour que cela marche). Elles prennent en compte ce que l'utilisateur introduit dans les entrées et insèrent du contenu dans certains widgets pour une meilleure ergonomie de l'interface. Les deux arguments en entrée sont, en effet, l'input saisi par l'utilisateur et le widget de résultat. Ces fonctions sont

par la suite utilisées dans des boutons pour répondre à la demande de l'utilisateur. Nous avons utilisé des fonctions anonymes pour appeler ces fonctions. Le fichier "Fonctions" contient aussi des fonctions étrangères à la partie 2 qui sont pratiques pour l'utilisation de l'interface.

Les principales difficultés de la partie 3 ont été de réussir à faire tourner correctement les fonctions dans le code de l'interface et de faire apparaître les résultats sur l'interface graphique. Pour le premier point, l'idée était de créer un module (Fonctions.py) afin d'alléger le code de l'interface. Cependant il ne suffisait pas d'importer les fonctions tels quelles, même avec des `.get()` les variables entrées par l'utilisateur n'étaient pas reconnues dans les fonctions du fait qu'elles ne se trouvaient plus dans le même script, c'est pourquoi on a procédé aux modifications mentionnées plus haut (introduction de paramètres, utilisation de fonctions anonymes). Suite à cela est venu le « second challenge » qui était d'afficher les résultats des fonctions directement sur l'interface, ce qui s'avère relativement pratique pour l'utilisateur. De base nos fonctions impliquaient des `print()` pour afficher les résultats, mais ces derniers communiquent avec la console python et non pas l'interface directement, la solution a été de créer des espaces adéquats sur l'interface permettant l'affichage de l'output, dans notre code ces espaces prennent la forme de widgets s'actualisant à l'aide d'`insert()` ou de `delete()`.

À partir des fonctions de la partie 2, trois modifications principales ont été apportées afin de visualiser les inputs et outputs dans l'interface :

- Au lieu de demander à l'utilisateur de saisir l'input dans la console, les fonctions utilisent des `.get()`, qui permettent de prendre en compte les inputs saisis directement dans les widgets, sont utilisées.
- Au lieu de `print()`, nous utilisons `.insert()` pour insérer du contenu dans des widgets de résultats avec `END` comme argument, afin d'ajouter les résultats à la fin du contenu déjà existant dans le widget (ici c'est le titre qu'on définira ultérieurement). De plus, pour le supprimer à chaque nouvelle saisie d'input par l'utilisateur, nous utilisons `.delete()` avec "2.0" (pour ne pas prendre en compte le titre) et `END` comme arguments.
- Pour la deuxième et troisième fonction, les résultats sont des dataframes. Par contre, nous reconnaissons que le dataframe affiché dans l'interface n'est pas bien centré et organisé. Malheureusement, nous n'arrivons pas à l'améliorer, donc nous décidons d'afficher seulement les titres et les artistes correspondants pour une meilleure lisibilité.

Concernant la mise en page de l'interface graphique, nous avons d'abord créé une fenêtre graphique grâce à la librairie Tkinter. Les manipulations les plus courantes ont été la création de *labels*, de *boutons* et d'*entrées* (souvent regroupées dans des *frames* nommés “boîtes”) que nous avons placés minutieusement dans la fenêtre. Les entrées servent à récupérer ce que l'utilisateur recherche, les boutons à effectuer des actions (le paramètre *command* appelle des fonctions) et les labels à orienter l'utilisateur sur l'interface.

Nous avons aussi inséré une image d'arrière plan pour une plus belle esthétique de l'interface.

Nous avons utilisé un widget Canvas pour y introduire une image que nous avons retouchée au préalable.

Aussi, nous avons introduit un bouton contenant une image permettant de se rendre directement sur le site Spotify via une petite fonction “f.link\_spotify” utilisant la méthode “*webbrowser.open(url)*”. Les images en questions sont disponibles au format .png dans le sous-dossier “Images”.

Concernant la première boîte (boite1 dans le code “Main”), lorsque l'utilisateur entre un nom d'artiste présent dans le dataframe, deux boutons sont présents, ceux-ci dépendent donc de la fonction Artiste\_info(). Le premier propose à l'utilisateur d'être redirigé vers la page Wikipedia de l'artiste dont il a saisi le nom. Pour cela, nous avons créé une fonction lien\_wikipedia qui utilise la dataframe artists. L'URL d'une page Wikipedia est donnée par : <https://fr.wikipedia.org/wiki/{}>, avec entre les accolades le nom de l'artiste sous la forme suivante : Taylor\_Swift. Il a fallu remplacer les espaces par des traits de soulignement, nous avons donc pour cela formaté l'URL avec *.replace()*. Nous avons aussi mis les premières lettres du nom des artistes en majuscule grâce à *.title()* (d'où la variable *nom\_artiste\_maj*). Si l'artiste n'a pas de page Wikipedia, l'utilisateur en sera informé. A noter que ces boutons n'amènent vers la page Wikipedia que si le nom de l'artiste qu'a saisi l'utilisateur est valide.

Après avoir créé le lien Wikipedia, nous avons trouvé intéressant d'ajouter une seconde fonction qui redirige vers la page Spotify de l'artiste saisi. Cette fonction suit la même logique que la précédente, mais elle se base sur l'ID de l'artiste.

Pour créer les widgets affichant les résultats de nos fonctions, nous avons utilisé le module scrolledtext de tkinter. Pour certaines fonctions, de nombreux résultats peuvent être trouvés, mais

nous ne voulions pas que le widget soit trop grand dans l'interface. Par conséquent, une barre de défilement était un bon choix. Nous utilisons `.insert()` avec "1.0" et un titre comme arguments pour informer les utilisateurs où les résultats souhaités peuvent être trouvés.

L'output de la fonction "*Artist\_info*" qui dépend du nom de l'artiste entré par l'utilisateur apparaît en haut à droite de l'interface dans un widget `res1`. C'est ici que l'utilisateur retrouve le nombre d'abonnés de l'artiste, ses chansons les plus populaires et les plus récentes ainsi que la présence ou non de musiques de l'artiste dans le top 200 mondial au cours de l'année 2020.

L'output de la fonction "*song*", qui dépend du nom de l'artiste entré par l'utilisateur, s'affiche en bas à droite de l'interface dans un widget `res2`, situé en dessous du widget `res1`. C'est ici que l'utilisateur retrouve jusqu'à 20 chansons possédant ce nom.

L'output de la fonction "*Angenre*", qui dépend de l'année et du genre musical entrés par l'utilisateur, s'affiche en bas à gauche de l'interface dans un widget `resu3`, positionné en dessous des widget `resu1` et `resu2`. C'est ici que l'utilisateur retrouve les chansons correspondant aux deux critères saisis.



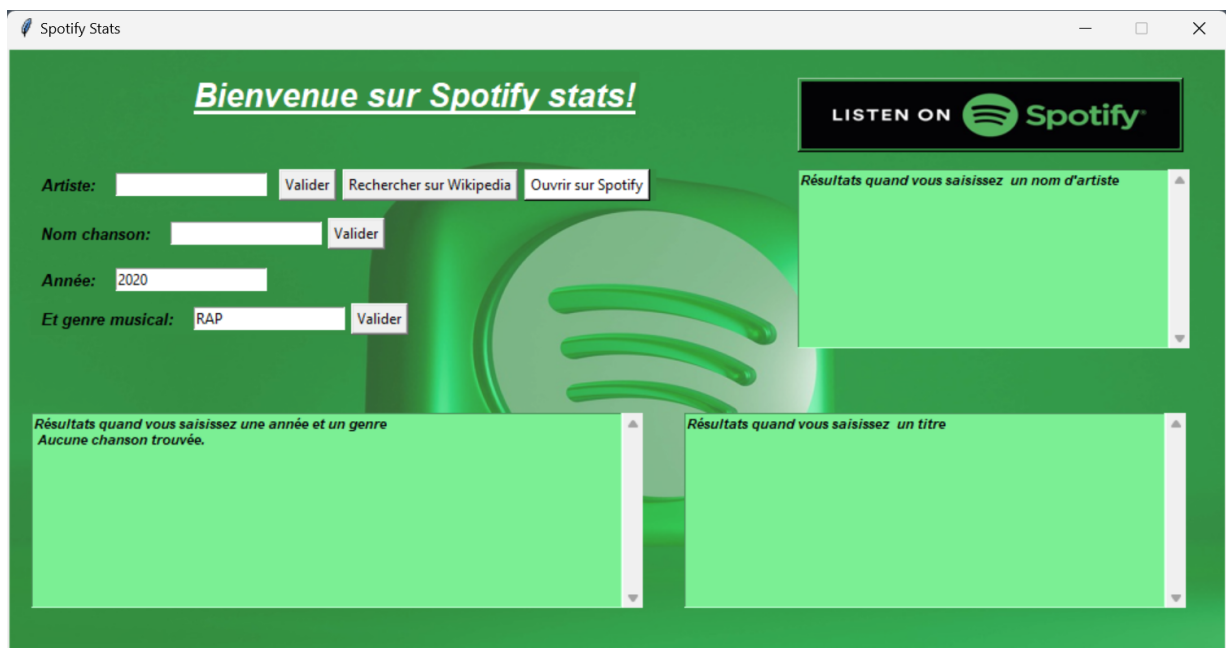
## COMMENT EXÉCUTER LE PROGRAMME ?

*Tous les chemins sont relatifs, vous aurez donc accès à tous les dataframes sans avoir à les changer. Les dossiers Partie <1, 2 ou 3> correspondent aux trois différentes parties du projet.*

1. Télécharger le dossier **Sujet\_1\_COLIN\_HOARAU\_SOP\_VO.zip** et extraire les fichiers qu'il contient.
2. Dans le dossier 'Partie 1', ouvrez le script **Projet\_Partie1**. Cette partie a été réalisée grâce à un notebook, il vous suffit d'exécuter les cellules dans l'ordre en n'oubliant surtout pas d'importer les différents packages au début.
3. Dans le dossier 'Partie 2', ouvrez le script **Projet\_Partie2**. Assurez-vous de bien importer toutes les librairies au début du code et exécutez toutes les lignes du nettoyage des données. Puis, vous pourrez exécuter chaque fonction l'une après l'autre et entrer l'input qui vous est demandé.
4. Dans le dossier 'Partie 3', ouvrez et exécutez les programmes dans l'ordre suivant : **traitement\_data.py**, **Fonctions.py**, **Main(interface).py**. Suite à cela, lorsque l'interface s'affiche, entrez l'une (ou les dans le cas de la troisième recherche) des informations demandées puis appuyez sur Valider. Si vous avez entré le nom de l'artiste, vous aurez la possibilité de rechercher cet artiste sur Wikipedia ou Spotify en appuyant sur leurs boutons respectifs.

## Nos difficultés...

- Nous avons eu l'idée d'ajouter un menu déroulant pour le genre de musique afin de faciliter la recherche lorsque l'on ne connaît pas tous les genres possibles. Cependant, par manque de temps nous avons dû laisser de côté cette idée, même si nous avions un début de code permettant de le faire (en utilisant *Combobox*). De plus, le menu n'aurait pas été complet puisque nous n'avons pas trouvé comment faire pour qu'il s'adapte en fonction de ce que l'utilisateur saisit.
- Nous souhaitons afficher les dataframes de manière plus lisible dans l'interface. Nous avons tenté avec Treeview (mais il n'affiche dans un autre onglet que l'interface, donc nous avons abandonné), pandastable, ainsi que panda GUI. Cependant, en raison des contraintes de temps, nous n'avons pas encore réussi à mettre en œuvre ces méthodes.
- Aussi, nous rencontrons un problème que nous n'avons pas eu le temps de résoudre concernant l'output de la requête sur l'année et le genre, ie la fonction "*Angenre*". En effet, lorsque l'on démarre l'interface graphique, si l'on cherche en premier des chansons correspondant à certains genres comme le Rap ou le Jazz (avec une année), il est écrit qu'il n'y a aucune chanson trouvée.



Pourtant, si l'on fait une recherche sur un nom de chanson (entrée 2) au préalable, et qu'on relance notre recherche initiale, alors la requête fonctionne correctement.

Spotify Stats

Bienvenue sur Spotify stats!

Artiste:  Valider


Rechercher sur Wikipedia

Ouvrir sur Spotify

Nom chanson:  Valider

Année:

Et genre musical:  Valider

LISTEN ON  Spotify

Résultats quand vous saisissez un nom d'artiste

Résultats quand vous saisissez une année et un genre

Voici les chansons trouvées:

Titre	Artistes
te mudaste	bad bunny
booker t	bad bunny
toosie slide	drake
haciendo que me amas	bad bunny
si veo a tu mamá	bad bunny
yo perreo sofa	bad bunny
la difícil	bad bunny
a tu merced	bad bunny
yo visto así	bad bunny

Résultats quand vous saisissez un titre

Voici les chansons trouvées qui correspondent à Sky:

Titre	Artistes
sky	playboi carti
sky	sonique
sky	yiruma
sky	blauw blume
sky	bennie k

## **Ce que le projet nous a appris...**

**Nhi :** Au cours de ma Licence, je n'ai jamais utilisé Python, donc je n'ai jamais vraiment codé en Python. Ce projet couvre les notions que j'ai abordées pendant le cours de Langage de Programmation et me permet d'explorer des aspects que je n'ai pas encore appris en cours, comme Tkinter. Au début, quand je lisais les questions du projet, je ne savais pas par où commencer car bien que je sache comment coder, je ne me suis pas encore familiarisé avec Python. En ce qui concerne Tkinter, au début, je n'ai pas totalement compris les concepts. Il y a eu des moments où je n'avais aucune idée de ce que je devais faire. Puis, j'ai appris comment découper les questions, déterminer l'ordre des codes à mettre, et interpréter les messages d'erreurs. Je ne sais plus combien d'erreurs j'ai eues en réalisant ce projet, mais je constate que j'apprends mieux à partir de mes erreurs. Cela m'énerve sûrement à chaque fois que j'ai une erreur, mais après des heures passées devant mon ordinateur, à chercher sur Internet et à demander aux camarades, je suis toujours contente de l'avoir résolue. Je ne suis toujours pas une pro en Python, mais mon niveau maintenant est nettement plus élevé qu'en septembre.

**Léo:** J'ai trouvé le sujet intéressant et le projet stimulant. N'ayant jamais programmé avant cette année, je suis content de ce que nous avons réussi à réaliser. J'ai beaucoup aimé le côté concret et appliqué du projet qui met en lumière la force de la programmation et qui nous permet d'imaginer tout ce qu'il est possible de faire avec. Aussi, je pense que le fait d'être en groupe est quelque chose d'enrichissant et de formateur. Cela nous apprend notamment à diviser les tâches et à communiquer pour travailler de manière efficace.

**Brice :** Ce projet m'a tout d'abord permis d'agréger une bonne partie des connaissances vues en cours afin de le mener à bien avec mon groupe. N'ayant jamais fait de programmation avant, c'est un aspect de la formation que je redoutais, mais ce projet m'a permis de voir les choses sous un autre angle que ce soit dans l'organisation du projet comme dans la résolution des problèmes auxquelles on a fait face. Maintenant si je devais citer une chose qui m'a été apprise via ce projet c'est la librairie Tkinter (pour le meilleur et pour le pire), je pensais pas qu'il était possible d'établir des interfaces graphiques interagissant avec les utilisateurs et malgré certaines manipulations qui ont été dures à élaborer je constate que j'ai pris goût à la construction de notre interface.

**Léa :** N'ayant jamais fait de programmation sur Python avant ce cours, ce projet m'a permis de mieux comprendre comment fonctionne ce langage de programmation. Il m'a permis en premier lieu d'appliquer certaines des notions vues en cours sur de vraies données et de les approfondir davantage, mais aussi par la suite de me rendre compte des possibilités multiples que propose ce langage, par exemple avec la découverte de la création d'interface graphique. Il a certes nécessité du temps, de la patience mais je peux maintenant confirmer que la meilleure façon de s'améliorer est de s'entraîner. En effet, le fait de chercher l'erreur dans un programme, d'essayer des changements, mais aussi de faire ses propres recherches sur Internet (chose que je ne faisais pas auparavant), c'est cela qui permet de progresser. De plus, le projet a pu être réalisé de manière plus efficace en groupe, il a fallu se coordonner et s'entraider pour arriver au meilleur résultat possible en l'espace de quelques semaines. Le projet m'a paru d'autant plus intéressant car il concerne une plateforme que j'utilise quotidiennement. Finalement, réaliser ce projet m'a permis de comprendre qu'il est normal de ne pas tout réussir du premier coup, ce que j'avais du mal à me dire lorsque j'effectuais les exercices de TD.

# INTERFACE JOUR 1



... ET

# INTERFACE JOUR - J

