

✓ Analisis de calidad de datos y reporte de resultados

Este proyecto se realiza en el marco de la candidatura de Henry Cardenas para la vacante como Data Quality Engineer para R5. Se proporcionó un JSON con informacion adquirida mediante la API de spotify, el cual fue normalizado en un CSV para realizar un análisis de calidad de datos.

Conjuntos de datos proporcionados:

Informacion de Spotify de Taylor Swift en formato CSV

Dimensiones del marco de calidad de datos:

Siguiendo la metodología planteada por DAMA UK se evaluarán las siguientes dimensiones de cada una de las variables de interes:

Compleitud: la proporción de datos almacenados respecto al "100%" de los datos.

Unicidad: ningún registro es almacenado mas de una vez.

Oportunidad: el grado en el que los datos representan la realidad en el tiempo requerido.

Validez: los datos estan representandos conforme a la sintaxis de su definición (formato, tipo, rango)

Exactitud: el grado en que los datos representan la realidad del objeto o evento descrito.

Coherencia: la ausencia de diferencia al comparar dos o más fuentes representaciones respecto a una definición.

```
# importando la libreria pandas
import pandas as pd

# cargando la informacion del CSV en un DataFrame
tsf_df = pd.read_csv("taylor_swift_spotify_flattened.csv")
```

✓ Explorando y analizando la calidad del dato de: Información de Spotify de Taylor Swift

```
#mostrar los datos contenidos en el DF
print(tsf_df.head())
```

	disc_number	duration_ms	explicit	track_number	track_popularity	\
0	1	212600	False	1	77	
1	1	231833	False	2	78	
2	1	231000	False	3	79	
3	1	235800	False	4	78	
4	1	193289	False	5	77	

	track_id	track_name	\
0	4WUepByoeqcedHoYhSNHRt	Welcome To New York (Taylor's Version)	
1	0108kcWLnn2H1H2kedi1gn	Blank Space (Taylor's Version)	
2	3Vpk1hfMAQme8VJ0SNRSkd	Style (Taylor's Version)	
3	10cSfkeCg9hRC2sFKB4IMJ	Out Of The Woods (Taylor's Version)	
4	2k0ZEeAqzvYMcx9Qt5aClQ	All You Had To Do Was Stay (Taylor's Version)	

	audio_features.danceability	audio_features.energy	audio_features.key	\
0	0.757	0.610	7.0	
1	0.733	0.733	0.0	
2	0.511	0.822	11.0	
3	0.545	0.885	0.0	
4	0.588	0.721	0.0	

	... audio_features.tempo	audio_features.id	\
0	... 116.998	4WUepByoeqcedHoYhSNHRt	
1	... 96.057	0108kcWLnn2H1H2kedi1gn	
2	... 94.868	3Vpk1hfMAQme8VJ0SNRSkd	
3	... 92.021	10cSfkeCg9hRC2sFKB4IMJ	
4	... 96.997	2k0ZEeAqzvYMcx9Qt5aClQ	

	audio_features.time_signature	artist_id	artist_name	\
0	4.0	06HL4z0CvFAxyc27GX	Taylor Swift	
1	4.0	06HL4z0CvFAxyc27GX	Taylor Swift	
2	4.0	06HL4z0CvFAxyc27GX	Taylor Swift	
3	4.0	06HL4z0CvFAxyc27GX	Taylor Swift	
4	4.0	06HL4z0CvFAxyc27GX	Taylor Swift	

	artist_popularity	album_id	\
0	120	1o59UpKw81iHR0HPiSkJR0	

```

1      120  1o59UpKw81iHR0HPiSkJR0
2      120  1o59UpKw81iHR0HPiSkJR0
3      120  1o59UpKw81iHR0HPiSkJR0
4      120  1o59UpKw81iHR0HPiSkJR0

      album_name album_release_date album_total_tracks
0  1989 (Taylor's Version) [Deluxe]      2023-10-27      22
1  1989 (Taylor's Version) [Deluxe]      2023-10-27      22
2  1989 (Taylor's Version) [Deluxe]      2023-10-27      22
3  1989 (Taylor's Version) [Deluxe]      2023-10-27      22
4  1989 (Taylor's Version) [Deluxe]      2023-10-27      22

[5 rows x 27 columns]

```

✎ Revisando la completitud de los datos

```

# revisar valores nulos
total_valores_nulos = tsf_df.isnull().sum()




# calcular el total de valores
total_valores = tsf_df.count().sort_values(ascending=True)

# calcular el porcentaje de valores nulos
porcentaje_de_valores_nulos = total_valores_nulos/total_valores *100

# convertir al dataframe de valores faltantes
completitud = pd.concat({'total valores' : total_valores, ' total valores nulos': total_valores_nulos, 'pocentaje de valores nulos (%)': po

# display missing values
completitud

```

	total valores	total valores nulos	pocentaje de valores nulos (%)	
album_name	477	62	12.997904	
track_id	531	8	1.506591	
track_name	532	7	1.315789	
audio_features.energy	537	2	0.372439	
audio_features.loudness	537	2	0.372439	
audio_features.danceability	537	2	0.372439	
audio_features.time_signature	538	1	0.185874	
audio_features.speechiness	538	1	0.185874	
audio_features.key	538	1	0.185874	
audio_features.acousticness	538	1	0.185874	
audio_features.liveness	538	1	0.185874	
audio_features.tempo	538	1	0.185874	
artist_name	539	0	0.000000	
artist_id	539	0	0.000000	
artist_popularity	539	0	0.000000	
album_id	539	0	0.000000	
audio_features.id	539	0	0.000000	
disc_number	539	0	0.000000	
audio_features.instrumentalness	539	0	0.000000	
album_release_date	539	0	0.000000	
audio_features.mode	539	0	0.000000	
track_popularity	539	0	0.000000	
track_number	539	0	0.000000	
explicit	539	0	0.000000	
duration_ms	539	0	0.000000	
audio_features.valence	539	0	0.000000	
album_total_tracks	539	0	0.000000	

```
# creando un dataframe con los id de album con el nombre como valor nulo
completitud_album_name = tsf_df[tsf_df["album_name"].isna()]
print(completitud_album_name["album_id"].drop_duplicates())
print(completitud_album_name["album_total_tracks"].drop_duplicates())
```

```
329    1MPAXuTVL2Ej5x0JHiSPq8
429    6fyR4wBPwLHKcRtxgd4sGh
Name: album_id, dtype: object
329    46
429    16
Name: album_total_tracks, dtype: object
```

la columna "album_name" presenta 62 valores nulos, correspondientes a las 62 pistas en los dos albums sin nombre con album_id "1MPAXuTVL2Ej5x0JHiSPq8" y "6fyR4wBPwLHKcRtxgd4sGh"

```
# creando un dataframe con los nombres de las pistas que tienen el id como valor nulo
completitud_track_id = tsf_df[tsf_df["track_id"].isna()]
print(completitud_track_id["track_name"].drop_duplicates())
```

```
321    Gorgeous
363    Jump Then Fall
375    Welcome To New York
379    All You Had To Do Was Stay
382    Bad Blood
434    Back To December/Apoloize/You're Not Sorry - ...
442    Enchanted - Live/2011
445    Mine - POP Mix
Name: track_name, dtype: object
```

la columna "track_id" presenta 8 valores nulos, los cuales se encuentran en las canciones identificadas mediante nombre e indice en el espacio anterior.

```
# creando un dataframe con los id de las pistas que tienen el nombre como valor nulo
completitud_track_name = tsf_df[tsf_df["track_name"].isna()]
print(completitud_track_name["track_id"].drop_duplicates())

77      1QQii3pa5m8MEda0nbkjfw
91      0Z2kkf2zMkwRGQjZ7T4p8f
104     7712gjoih4QoDbXpljEk21
391     4FoV729rw7IhoK1MZW5K5V
396     71PmZqBXH0RUETqxpwlV0w
401     0TvQLMecTE8utz0NmvXRbK
408     7gJtmLyPTwKzhGzMBXuXH
Name: track_id, dtype: object
```

la columna "track_name" presenta 7 valores nulos, los cuales se encuentran en las canciones identificadas mediante "Track_id" e indice en el espacio anterior.

```
# creando un dataframe para cada "audio feature" con valores nulos.
completitud_af_energy = tsf_df[tsf_df["audio_features.energy"].isna()]
completitud_af_loudness = tsf_df[tsf_df["audio_features.loudness"].isna()]
completitud_af_danceability = tsf_df[tsf_df["audio_features.danceability"].isna()]
completitud_af_time_signature = tsf_df[tsf_df["audio_features.time_signature"].isna()]
completitud_af_speechiness = tsf_df[tsf_df["audio_features.speechiness"].isna()]
completitud_af_key = tsf_df[tsf_df["audio_features.key"].isna()]
completitud_af_acousticness = tsf_df[tsf_df["audio_features.acousticness"].isna()]
completitud_af_liveness = tsf_df[tsf_df["audio_features.liveness"].isna()]
completitud_af_tempo = tsf_df[tsf_df["audio_features.tempo"].isna()]

# creando una lista de dataframes para usar en la concatenacion.
afs = [completitud_af_energy,
completitud_af_loudness,
completitud_af_danceability,
completitud_af_time_signature,
completitud_af_speechiness,
completitud_af_key,
completitud_af_acousticness,
completitud_af_liveness,
completitud_af_tempo]
# creando una lista de columnas para llamar la informacion del dataframe concatenado
afs_cols = ["track_name", "audio_features.energy", "audio_features.loudness", "audio_features.danceability", "audio_features.time_signature",
            "audio_features.speechiness", "audio_features.key", "audio_features.acousticness", "audio_features.liveness", "audio_features.tempo"]

# concatenando los data frames
afs_resume = pd.concat(afs)

# removiendo los duplicados para mostrar la tabla de resumen de las canciones con valores de "audio features" nulos
afs_resume[afs_cols].drop_duplicates()
```

	track_name	audio_features.energy	audio_features.loudness	audio_features.danceability	audio_features.time_signature	audio_feat
330	Wildest Dreams	NaN	NaN	NaN	4.0	
363	Jump Then Fall	NaN	-5.712	0.617	NaN	
334	Teardrops On My Guitar - Radio Single Remix	0.427	NaN	0.626	4.0	
431	The Story Of Us - Live	0.908	-5.156	NaN	4.0	
341	So It Goes...	0.610	-7.283	0.574	4.0	
432	Mean - Live/2011	0.915	-4.373	0.429	4.0	

En la tabla anterior se pueden apreciar los registros de "audio_features" que se encuentran sin datos con el nombre de su respectiva canción.

Las demas columnas no presentan valores nulos

✓ Revisando la unicidad de los datos

```
# revisar valores duplicados en el dataframe principal
valores_duplicados = tsf_df.duplicated()

# calcular e imprimir el numero de registros duplicados y la unicidad porcentual del data frame
print("el numero de registros duplicados en el dataframe es {}".format(valores_duplicados.sum()))
print("su unicidad es del {} %".format(round((total_valores["album_id"]-valores_duplicados.sum())/total_valores["album_id"]*100,4)))

    el numero de registros duplicados en el dataframe es 18
    su unicidad es del 96.6605 %

# mostrar la tabla de valores duplicados en el dataframe principal organizados por "track_number" para darle detalle al informe
tsf_df[valores_duplicados].sort_values("track_number")
```

	disc_number	duration_ms	explicit	track_number	track_popularity	track_id	track_name	audio_features.danceabil
295	1	170640	False	1	77	43rA71bccXFGD4C8GOplIN	I Forgot That You Existed	0.1
297	1	221306	False	3	92	1dGr1c8CrMLDpV6mPblmSI	Lover	0.1
298	1	190360	False	4	86	3RauEVgRgj1luWdJ9fDs70	The Man	0.1
299	1	211240	False	5	82	3pHkh7d0IzM2AldUtz2x37	The Archer	0.1
300	1	173386	False	6	78	2YWtcWi3a83pdEg3Gif4Pd	I Think He Knows	0.1
301	1	234146	False	7	83	214nt20w5wOxJnY462klLw	Miss Americana & The Heartbreak Prince	0.1
302	1	222400	False	8	86	4y5bvROuBDPr5fuwXbIBZR	Paper Rings	0.1
303	1	287266	False	9	81	12M5uqx0ZuwkpLp5rJim1a	Cornelia Street	0.1
304	1	198533	False	10	79	2dgFqt3w9xIQRjhPtwnK3D	Death By A Thousand Cuts	0.1
305	1	190240	False	11	80	1LLXZFeAHK9R4xUramtUKw	London Boy	0.1
306	1	201586	False	12	72	4AYtqFyFbX0Xkc2wtcygTr	Soon You'll Get Better (feat. The Chicks)	0.1
307	1	200306	False	13	78	5hQSXkFgbxjZo9uCwd11so	False God	0.1
308	1	171360	False	14	84	6RRNNciQGZEXnqk8SQ9yv5	You Need To Calm Down	0.1
309	1	223293	False	15	82	1SymEzIT3H8UZfibCs3TYi	Afterglow	0.1
310	1	193000	False	16	80	2Rk4JINc2TPmZe2af99d45	ME! (feat. Brendon Urie of Panic! At The Disco)	0.1
311	1	150440	False	17	72	1SmiQ65iSAbPto6gPFIBYm	It's Nice To Have A Friend	0.1
312	1	293453	False	18	85	1fzAuUVbzlhZ1IJAx9PtY6	Daylight	0.1
88	1	234466	True	21	82	3xYJScVfxByb61dYHTwiby	Hits Different	0.1

18 rows × 27 columns

Dentro de los 18 registros identificados como duplicados se encuentran 17 canciones del álbum "lovers" exceptuando la pista 2 "Cruel Summer" que no se identifica como duplicada en el dataframe general debido a que para la columna "explicit" el duplicado tiene el valor "no" en vez de "false", lo cual será retomado al revisar la validez de la columna. También se identifica la canción "Hits Different" como registro duplicado apareciendo como track numero 24 en el álbum "Midnights (The Til Dawn Edition)" el cual en efecto reporta 24 pistas.

✓ Revisando la Oportunidad de los datos

El JSON fue creado en google drive el 26 dic 2023 a las 20:45 por el usuario L. C., sin embargo teniendo en cuenta que la información es estática y la evaluación de esta dimensión es opcional se asumirá que es oportuna.

✓ Revisando la Validez de los datos

Para verificar la validez de los datos se contrastará cada columna con el tipo de dato y su formato esperado teniendo en cuenta la documentación de la API de spotify.

```
# obteniendo informacion sobre tipo de dato y numero de valores nulos en el dataframe principal.
tsf_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 539 entries, 0 to 538
Data columns (total 27 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   disc_number                             539 non-null    int64
1   duration_ms                             539 non-null    int64
2   explicit                                 539 non-null    object
3   track_number                             539 non-null    int64
4   track_popularity                         539 non-null    int64
5   track_id                                 531 non-null    object
6   track_name                               532 non-null    object
7   audio_features.danceability              537 non-null    float64
8   audio_features.energy                    537 non-null    float64
9   audio_features.key                       538 non-null    float64
10  audio_features.loudness                   537 non-null    float64
11  audio_features.mode                       539 non-null    int64
12  audio_features.speechiness                538 non-null    float64
13  audio_features.acousticness               538 non-null    float64
14  audio_features.instrumentalness           539 non-null    object
15  audio_features.liveness                   538 non-null    float64
16  audio_features.valence                    539 non-null    float64
17  audio_features.tempo                      538 non-null    float64
18  audio_features.id                         539 non-null    object
19  audio_features.time_signature             538 non-null    float64
20  artist_id                                 539 non-null    object
21  artist_name                               539 non-null    object
22  artist_popularity                         539 non-null    int64
23  album_id                                 539 non-null    object
24  album_name                               477 non-null    object
25  album_release_date                       539 non-null    object
26  album_total_tracks                       539 non-null    object
dtypes: float64(10), int64(6), object(11)
memory usage: 113.8+ KB
```

✓ disc_number

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-track>" el disc_number presenta la siguiente descripción:

disc_number integer

The disc number (usually 1 unless the album consists of more than one disc).

```
# contar el numero de valores distintos para la columna
print(tsf_df.groupby("disc_number")["disc_number"].count())
```

```
disc_number
1      522
2       17
Name: disc_number, dtype: int64
```

Para la columna se puede apreciar que los valores son 1 o 2 siendo 1 el más común como se refleja en la documentación de la API.

✓ duration_ms

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-track>" la "duration_ms" presenta la siguiente descripción:

duration_ms integer

The track length in milliseconds.

```
# generar estadísticas descriptivas de la variable numerica.
tsf_df["duration_ms"].describe()

count      539.000000
mean      236003.725417
std       55019.871010
min      -223093.000000
25%      209486.500000
50%      233626.000000
75%      259045.500000
max       613026.000000
Name: duration_ms, dtype: float64
```

Se pueden apreciar valores negativos, lo cuál es imposible para una variable temporal, a continuación se presentan los registros con valores inferiores a 1000 con el fin de identificar también pistas inusualmente cortas.

```
# filtrando el dataframe principal para "duration_ms" menor que 60000
tsf_df[tsf_df["duration_ms"]<60000]
```

	disc_number	duration_ms	explicit	track_number	track_popularity	track_id	track_name	audio_features.danceabili
392	1	-107133	False	18	0	4eTXfpHxhxVofrBUjAhPMg	I Wish You Would - Voice Memo	0.7
408	1	-223093	False	2	59	7gJtmLyPTwKzhGzMBXtuXH	NaN	0.6
420	1	10	False	14	56	5PjfMmF06QtxTPZBZHdhoZ	Everything Has Changed	0.6
432	1	1000	False	4	47	7mFiEij8AXPUZB7aKLbUIQ	Mean - Live/2011	0.4
472	1	3000	False	14	152	7BFc7ffruhZ4Hecnqf5xju	Long Live	0.4

5 rows × 27 columns

Es posible identificar 5 registros con duraciones inusuales, dentro de los cuales existen dos negativos, y tres de menos de 1 minuto.

Explicit

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-track>" la "explicit" presenta la siguiente descripción:

explicit

boolean

Whether or not the track has explicit lyrics (true = yes it does; false = no it does not OR unknown).

```
# contar el número de valores distintos para la columna
print(tsf_df.groupby("explicit")["explicit"].count())

explicit
False    480
No         4
Si         1
True      54
Name: explicit, dtype: int64
```

La columna "explicit" debería ser un booleano solo con valores "True" o "False", sin embargo presenta 4 registros en el valor "No" y 1 registro en el valor "Si"

track_number

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-track>" la columna "track_number" presenta la siguiente descripción:

track_number integer

The number of the track. If an album has several discs, the track number is the number on the specified disc.

```
# mostrar el tipo de variable de la columna
print(tsf_df["track_number"].dtype)
# generar estadísticas descriptivas de la variable numérica.
tsf_df["track_number"].describe()

int64
count    539.000000
mean      11.280148
std        7.965621
min         1.000000
25%         5.000000
50%        10.000000
75%        15.000000
max        46.000000
Name: track_number, dtype: float64
```

La columna "track_number" es de tipo int64 con mínimo 1 y máximo 46 teniendo valores dentro de los esperados.

✓ track_popularity

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-track>" la columna "track_popularity" presenta la siguiente descripción:

popularity integer

The popularity of the track. The value will be between 0 and 100, with 100 being the most popular.

The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. Generally speaking, songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. Duplicate tracks (e.g. the same track from a single and an album) are rated independently. Artist and album popularity is derived mathematically from track popularity. **Note:** the popularity value may lag actual popularity by a few days: the value is not updated in real time.

```
# generar estadísticas descriptivas de la variable numérica.
tsf_df["track_popularity"].describe()

count    539.000000
mean      62.918367
std       22.498757
min     -92.000000
25%      51.000000
50%      69.000000
75%      77.000000
max     152.000000
Name: track_popularity, dtype: float64
```

Según su descripción en la documentación la columna "track_popularity" debería tener valores entre 0 y 100, sin embargo en la estadística descriptiva se aprecian valores fuera de este rango, estos valores se reportan en la tabla presentada en el siguiente bloque de código.

```
# filtrando el dataframe principal para "track_popularity" menor que 0 o mayor que 100 y concatenando los dataframes
tp_1 = tsf_df[tsf_df["track_popularity"]<0]
tp_2 = tsf_df[tsf_df["track_popularity"]>100]
pd.concat([tp_1, tp_2])
```

	disc_number	duration_ms	explicit	track_number	track_popularity	track_id	track_name	audio_features.danceabil
75	1	204852	True	11	-69	45R112Jz5hQeKgITXgSXzs	Karma	0.
89	1	202395	True	1	-70	4g2c7NoTWAOSYDy44l9nub	Lavender Haze	0.
109	1	202395	True	1	-85	5jQl2r1RdgtuT8S3iG8zFC	Lavender Haze	0.
111	1	200690	False	3	-92	0V3wPSX9ygBnCm8psDlegu	Anti-Hero	0.
115	1	210556	True	7	-75	0heeNYlwOGuUSe7TgUD27B	Question...?	0.
128	1	244586	False	7	-71	2r9CbjYgFhtAmcFv1cSquB	I Almost Do (Taylor's Version)	0.
472	1	3000	False	14	152	7BFc7ffruhZ4Hecnqf5xju	Long Live	0.

7 rows × 27 columns

Como se puede apreciar existen seis valores negativos para la columna y un valor por encima de 100, lo cual los convertiría en valores atípicos para la descripción.

track_id

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-track>" la columna "track_id" presenta la siguiente descripción:

id

string

Required

The [Spotify ID](#) for the track.

Example: `11dFghVXANMlKmJXsNCbNl`

```
# usando el primer registro que se encuentra validado para tomar la longitud del string
str_len = len(tsf_df['track_id'][0])
print("la longitud del string de la columna es {}".format(str_len))
# revisando que todos los registros de la columna sean strings de la misma longitud
tsf_df[tsf_df['track_id'].str.len()!=str_len]
```

la longitud del string de la columna es 22

	disc_number	duration_ms	explicit	track_number	track_popularity	track_id	track_name	audio_features.danceability
321	1	209680	False	8	84	NaN	Gorgeous	0.800
363	1	238093	False	35	32	NaN	Jump Then Fall	0.617
375	1	212600	False	1	60	NaN	Welcome To New York	0.789
379	1	193293	False	5	60	NaN	All You Had To Do Was Stay	0.605
382	1	211933	False	8	61	NaN	Bad Blood	0.646
434	1	362826	False	6	54	NaN	Back To December/Apologize/You're Not Sorry - ...	0.374
442	1	389213	False	14	49	NaN	Enchanted - Live/2011	0.340
445	1	230546	False	1	49	NaN	Mine - POP Mix	0.696

8 rows × 27 columns

Para la columna "track_id" el formato de string de 22 caracteres se cumple para todo los registros no nulos.

▼ track_name

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-track>" la columna "track_name" presenta la siguiente descripción:

name string

The name of the track.

```
# definir una funcion para aplicar a cada fila verificando si es un string.
def is_string(x):
    return isinstance(x, str)
# aplicar la funcion "is_string" a cada fila del dataframe y crear una columna nueva con el booleano correspondiente
tsf_df["track_name_str"] = tsf_df["track_name"].apply(is_string)
# seleccionar los registros con string a False
tsf_df[tsf_df["track_name_str"] == False]
```

	disc_number	duration_ms	explicit	track_number	track_popularity	track_id	track_name	audio_features.danceabil
77	1	191039	False	13	68	1QQii3pa5m8MEda0nbkjfw	NaN	0.
91	1	200690	False	3	71	02Zkkf2zMkwRGQjZ7T4p8f	NaN	0.
104	1	196258	False	16	72	7712gjoih4QoDbXpljEk21	NaN	0.
391	1	216333	False	17	0	4FoV729rw7IhoKIMZW5K5V	NaN	0.
396	1	231000	False	3	64	71PmZqBXH0RUETqxpwlV0w	NaN	0.
401	1	211933	False	8	79	0TvQLMecTE8utzoNmvXRbK	NaN	0.
408	1	-223093	False	2	59	7gJtmLyPTwKzhGzMBXtuXH	NaN	0.

7 rows × 28 columns

Se evidencia para la columna "track_name" que los únicos valores que no son string son los siete correspondientes a los valores nulos identificados anteriormente.

▼ audio_features

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-audio-features>" las columnas referentes a las "audio_features" se definen así:

danceability number [float]

Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

Example: 0.585

energy number [float]

Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

Example: 0.842

key integer

The key the track is in. Integers map to pitches using standard [Pitch Class notation](#). E.g. 0 = C, 1 = C#/D♭, 2 = D, and so on. If no key was detected, the value is -1.

Range: -1 - 11

Example: 9

loudness number [float]

The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.

Example: -5.883

mode integer

Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

Example: 0

speechiness number [float]

Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

Example: 0.0556

acousticness number [float]

A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

Range: 0 - 1

Example: 0.00242

instrumentalness number [float]

Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

Example: 0.00686

liveness number [float]

Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

Example: 0.0866

valence number [float]

A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

Range: 0 - 1

Example: 0.428

tempo number [float]

The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

Example: 118.211

id string

The Spotify ID for the track.

Example: "2takcwOaAZWiXQijPHIx7B"

time_signature integer

An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".

Range: 3 - 7

Example: 4

```
# generar estadísticas descriptivas de la variable numérica.
tsf_df[["audio_features.danceability", "audio_features.energy", "audio_features.key",
"audio_features.loudness", "audio_features.mode", "audio_features.speechiness", "audio_features.acousticness"]].describe()
```

	audio_features.danceability	audio_features.energy	audio_features.key	audio_features.loudness	audio_features.mode	audio_features.speechiness
count	537.000000	537.000000	538.000000	537.000000	539.000000	537.000000
mean	0.587242	0.573065	4.587361	-7.520639	0.912801	0.161993
std	0.116858	0.192309	3.246082	2.933158	0.282388	0.045130
min	0.243000	0.118000	0.000000	-17.932000	0.000000	0.000000
25%	0.517000	0.436000	2.000000	-9.287000	1.000000	0.000000
50%	0.595000	0.589000	5.000000	-6.942000	1.000000	0.000000
75%	0.661000	0.729000	7.000000	-5.376000	1.000000	0.000000
max	0.897000	0.949000	11.000000	-1.909000	1.000000	0.000000

Debido a que se identifica anteriormente que la columna "audio_features.instrumentalness" es de tipo "object" se presume que contiene strings, por lo que se revisará más en detalle.

```
# verificar el tipo de datos en la columna
tsf_df["audio_features.instrumentalness"].dtypes

dtype('O')
```

Como efectivamente el tipo es objeto se procede a intentar convertirlo a tipo "float" que sería propio de las dimensiones y límites establecidos en la documentación.

```
# convertir la columna a tipo float
tsf_df["audio_features.instrumentalness"].astype(float)
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-32-223992d200f3> in <cell line: 2>()
      1 # convertir la columna a tipo float
----> 2 tsf_df["audio_features.instrumentalness"].astype(float)

----- 6 frames -----
/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in astype_nansafe(arr, dtype, copy, skipna)
    168     if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
    169         # Explicit copy, or required since NumPy can't view from / to object.
--> 170         return arr.astype(dtype, copy=True)
    171
    172     return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: '7.28x-06'
```

EXPLICAR ERROR

El intento de convertir la columna a tipo float arroja un error debido a la presencia del valor "7.28x-06" el cuál al contener una "x" en lugar de la "e" convencional de la notación científica genera el error.

```
# creando un dataframe reemplazando "x" para probar la conversion nuevamente e identiicar valores adicionales que generen error
conversion_test = tsf_df["audio_features.instrumentalness"].str.replace("x","e")
```



```
conversion_test.astype(float)

0      0.000037
1      0.000000
2      0.019700
3      0.000056
4      0.000000
...
534    0.000000
535    0.000807
536    0.000000
537    0.000000
538    0.000000
Name: audio_features.instrumentalness, Length: 539, dtype: float64
```

Como se puede observar al remover el valor los demás valores son convertidos a float sin inconveniente.

Continuando con las demás "audio_features"

```
# generar estadísticas descriptivas de la variable numerica.
tsf_df[["audio_features.liveness", "audio_features.valence", "audio_features.tempo",
"audio_features.id", "audio_features.time_signature"]].describe()
```

	audio_features.liveness	audio_features.valence	audio_features.tempo	audio_features.time_signature	
count	538.000000	539.000000	538.000000	538.000000	
mean	0.163308	0.398410	122.362639	3.986989	
std	0.141800	0.199409	30.485522	0.197323	
min	0.033500	0.037400	68.097000	3.000000	
25%	0.096500	0.230000	96.684500	4.000000	
50%	0.115000	0.386000	119.000500	4.000000	
75%	0.162250	0.535000	143.939000	4.000000	
max	0.931000	0.943000	208.918000	5.000000	

Los valores numéricos se encuentran en los rangos esperados para cada una de las "audio_features", a continuación se revisará la condición del "audio_features.id" usando nuevamente el criterio de la columna "track_id"

```
# usando el primer registro que se encuentra validado para tomar la longitud del string
str_len = len(tsf_df['audio_features.id'][0])
print("la longitud del string de la columna es {}".format(str_len))
# revisando que todos los registros de la columna sean strings de la misma longitud
tsf_df[tsf_df['audio_features.id'].str.len()!=str_len]

la longitud del string de la columna es 22
disc_number duration_ms explicit track_number track_popularity track_id track_name audio_features.danceability audio_features.
0 rows x 29 columns
```

La propiedad cumple también con el string de longitud 22, además cuenta con los id completos en contraste con los valores nulos presentes en "track_id"

▼ artist

Los datos referente al artista estan dados como valores únicos en el nivel exterior del JSON, estos valores fueron replicados a toda la tabla para propósitos de evaluación posterior de los datos, sin embargo serán evaluados a continuacion:

```
print(tsf_df["artist_id"].unique())
print(tsf_df["artist_name"].unique())
print(tsf_df["artist_popularity"].unique())

['06HL4z0CvFAxyc27GX']
['Taylor Swift']
[120]
```

Como se puede apreciar los valores únicos corresponden a los valores replicados en el dataframe.

▼ album

Según la documentación encontrada en la url "<https://developer.spotify.com/documentation/web-api/reference/get-an-album>" las columnas referentes a las propiedades del "album" se definen asi:

id string Required

The [Spotify ID](#) of the album.

Example: `4aawyAB9vmqN3uQ7FjRGTy`

name string Required

The name of the album. In case of an album takedown, the value may be an empty string.

release_date string Required

The date the album was first released.

Example: `"1981-12"`

total_tracks integer Required

The number of tracks in the album.

Example: `9`

▼ album_id

Este se validará utilizando el metodo usado anteriormente con los id, reconociendo la longitud del string usado como identificador.

```
# usando el primer registro que se encuentra validado para tomar la longitud del string
str_len = len(tsf_df['album_id'][0])
print("la longitud del string de la columna es {}".format(str_len))
# revisando que todos los registros de la columna sean strings de la misma longitud
tsf_df[tsf_df['album_id'].str.len()!=str_len]

la longitud del string de la columna es 22
disc_number duration_ms explicit track_number track_popularity track_id track_name audio_features.danceability audio_features.
0 rows x 29 columns
```

Se aprecia que la columna "album_id" cuenta con todos los registros y a su vez todos ellos tienen 22 caracteres.

album_name

De manera similar a el "track_name" para esta columna se validará que cada entrada sea un string.

```
# definir una funcion para aplicar a cada fila verificando si es un string.
def is_string(x):
    return isinstance(x, str)
# aplicar la funcion "is_string" a cada fila del dataframe y crear una columna nueva con el booleano correspondiente
tsf_df["album_name_str"] = tsf_df["album_name"].apply(is_string)
# seleccionar los registros con string a False
tsf_df[tsf_df["album_name_str"] == False]
```

	disc_number	duration_ms	explicit	track_number	track_popularity	track_id	track_name	audio_features.danceability
329	1	329160	False	1	42	00vJzaoxM3Eja1doBUhX0P	All Too Well	0.6
330	1	220440	False	2	45	22C0JIVhFaczZ4t9heqREN	Wildest Dreams	N
331	1	246306	False	3	36	7APTsjmZbj7TFXQJAIrti4	The Best Day	0.6
332	1	223093	False	4	39	4V9NuhKQcUft4cgbynHV79	Red	0.6
333	1	202960	False	5	36	4hCZk6WJBfQeG3P7WsyntI	Holy Ground	0.6
...
440	1	83253	False	12	50	4KB3zBArZ1rHRExilzycaV	I Want You Back - Live/2011	0.5
441	1	404680	False	13	48	4DbI1rr4IQ2bc8nejy9ttD	Dear John - Live/2011	0.5
442	1	389213	False	14	49	NaN	Enchanted - Live/2011	0.3
443	1	292426	False	15	48	6znB7YzhMJT2B83zfbiyLq	Haunted - Live/2011	0.3
444	1	376466	False	16	48	5QkzyLUb4ggojNDp8G2OxX	Long Live - Live/2011	0.5

62 rows x 30 columns

Al revisar los valores de la columna se reportan los 62 valores nulos identificados anteriormente.

album_release_date

Es un string de fechas con formato "aaaa-mm-dd"

```
unique_relese_dates = tsf_df['album_release_date'].sort_values()
print(unique_relese_dates.unique())

['1989-10-24' '2008-06-28' '2008-11-11' '2010-01-01' '2010-10-25'
 '2012-10-22' '2014-01-01' '2017-11-09' '2017-11-10' '2019-08-23'
 '2020-07-24' '2020-08-18' '2020-11-25' '2020-12-11' '2021-01-07'
 '2021-04-09' '2021-11-12' '2022-10-21' '2022-10-22' '2023-07-07'
 '2023-10-26' '2023-10-27' '2027-05-26']
```