

Descripción de la columna DATASET SPOTIFY

track_id: El ID de Spotify de la canción

artists: Los nombres de los artistas que interpretaron la canción. Si hay más de un artista, se separan por un ;

album_name: El nombre del álbum en el que aparece la canción

track_name: Nombre de la canción

popularity: La popularidad de una pista es un valor entre 0 y 100, siendo 100 la más popular. La popularidad se calcula mediante un algoritmo y se basa, en su mayor parte, en el número total de reproducciones que ha tenido la canción y lo recientes que son esas reproducciones. En general, las canciones que se reproducen mucho ahora tendrán una mayor popularidad que las canciones que se reprodujeron mucho en el pasado. Las canciones duplicadas (por ejemplo, la misma canción de un single y de un álbum) se valoran de forma independiente. La popularidad de artistas y álbumes se obtiene matemáticamente a partir de la popularidad de las canciones.

duration_ms: La duración de la pista en milisegundos

explicit: Si la canción tiene o no letra explícita (verdadero = sí la tiene; falso = no la tiene O desconocido)

danceability: La bailabilidad describe lo adecuada que es una pista para bailar basándose en una combinación de elementos musicales que incluyen el tempo, la estabilidad del ritmo, la fuerza del compás y la regularidad general. Un valor de 0,0 es el menosailable y 1,0 el másailable.

energy: La energía es una medida de 0,0 a 1,0 y representa una medida perceptiva de intensidad y actividad. Normalmente, las pistas energéticas son rápidas, ruidosas y ruidosas. Por ejemplo, el death metal tiene mucha energía, mientras que un prelude de Bach puntúa bajo en la escala.

key: La tonalidad de la canción. Los números enteros se asignan a los tonos utilizando la notación estándar Pitch Class. Por ejemplo, 0 = Do, 1 = Do#/D♭, 2 = Re, y así sucesivamente. Si no se detecta ninguna tonalidad, el valor es -1

loudness: La sonoridad general de una pista en decibelios (dB)

mode: El modo indica la modalidad (mayor o menor) de una pista, el tipo de escala del que se deriva su contenido melódico. Mayor se representa con 1 y menor con 0

speechiness: La locuacidad detecta la presencia de palabras habladas en una pista. Cuanto más exclusivamente hablada sea la grabación (por ejemplo, programa de entrevistas, audiolibro, poesía), más se acercará a 1,0 el valor del atributo. Los valores superiores a 0,66 describen pistas que probablemente estén compuestas en su totalidad por palabras habladas. Los valores entre 0,33 y 0,66 describen pistas que pueden contener tanto música como voz, ya sea en secciones o en capas, incluyendo casos como la música rap. Los valores por debajo de 0,33 representan probablemente música y otras pistas no habladas.

acousticness: Medida de confianza de 0,0 a 1,0 para determinar si la pista es acústica. 1,0 representa una confianza alta en que la pista es acústica.

instrumentalness: Predice si una pista no contiene voces. Los sonidos "ooh" y "aah" se consideran instrumentales en este contexto. Las pistas de rap o spoken word son claramente "vocales". Cuanto más se acerque el valor de instrumental a 1,0, mayor será la probabilidad de que la pista no contenga voces.

liveness: Detecta la presencia de público en la grabación. Los valores más altos representan una mayor probabilidad de que la pista haya sido interpretada en directo. Un valor superior a 0,8 indica una gran probabilidad de que la pista se haya grabado en directo.

valance: Medida de 0,0 a 1,0 que describe la positividad musical que transmite una pista. Las pistas con una valencia alta suenan más positivas (por ejemplo, felices, alegres, eufóricas), mientras que las pistas con una valencia baja suenan más negativas (por ejemplo, tristes, deprimidas, enfadadas).

tempo: el tempo global estimado de una pista en pulsaciones por minuto (BPM). En terminología musical, el tempo es la velocidad o el ritmo de una pieza determinada y se deriva directamente de la duración media de los tiempos.

time_signature: Un compás estimado. El compás es una convención para especificar cuántos tiempos hay en cada compás. El compás oscila entre 3 y 7, indicando compases de 3/4 a 7/4.

track_genre: El género al que pertenece la pista

El conjunto de datos se distribuye en: **114000 Registros o filas y 21 columnas.**

```
¿Cuántas variables y observaciones  
tenemos en el conjunto de datos?
```

```
1 df.shape  
Executed at 2023.10.14 22:19:10 in 160ms
```

```
(114000, 21)
```

Observamos que solo hay 21 columnas, las cuales 15 **columnas son numericas**, 5 **categoricas** y una **booleana** .

```
¿Cuántas variables de cada tipo de dato tenemos en el conjunto de datos?
```

```
(
  df
  .dtypes
  .value_counts()
)
```

Executed at 2023.10.14 22:19:10 in 176ms

< 4 rows > | Length: 4, dtype: int64 pd.Series

	<unnamed>
float64	9
int64	6
object	5
bool	1

Tenemos 3 columnas con datos nulos.

- album_name
- artist
- track_name

```
De tener observaciones con valores nulos, ¿cuántas tenemos por cada variable?
```

```
1 (
2   df
3   .isnull()
4   .sum()
5   .sort_values(ascending=False)
6 )
7
```

Executed at 2023.10.14 22:19:11 in 143ms

< 1-10 > | Length: 21, dtype: int64 pd.Series

	<unnamed>
artists	1
album_name	1
track_name	1
Unnamed: 0	0
mode	0
time_signature	0
tempo	0

Podemos ver de que esto solo pertenece a un solo registro

```
# Para obtener todas las filas con valores nulos:
rows_with_null_values = df[df.isnull().any(axis=1)]
# Luego puedes mostrar o trabajar con estas filas:
print(rows_with_null_values)
```

Executed at 2023.10.14 23:04:39 in 171ms

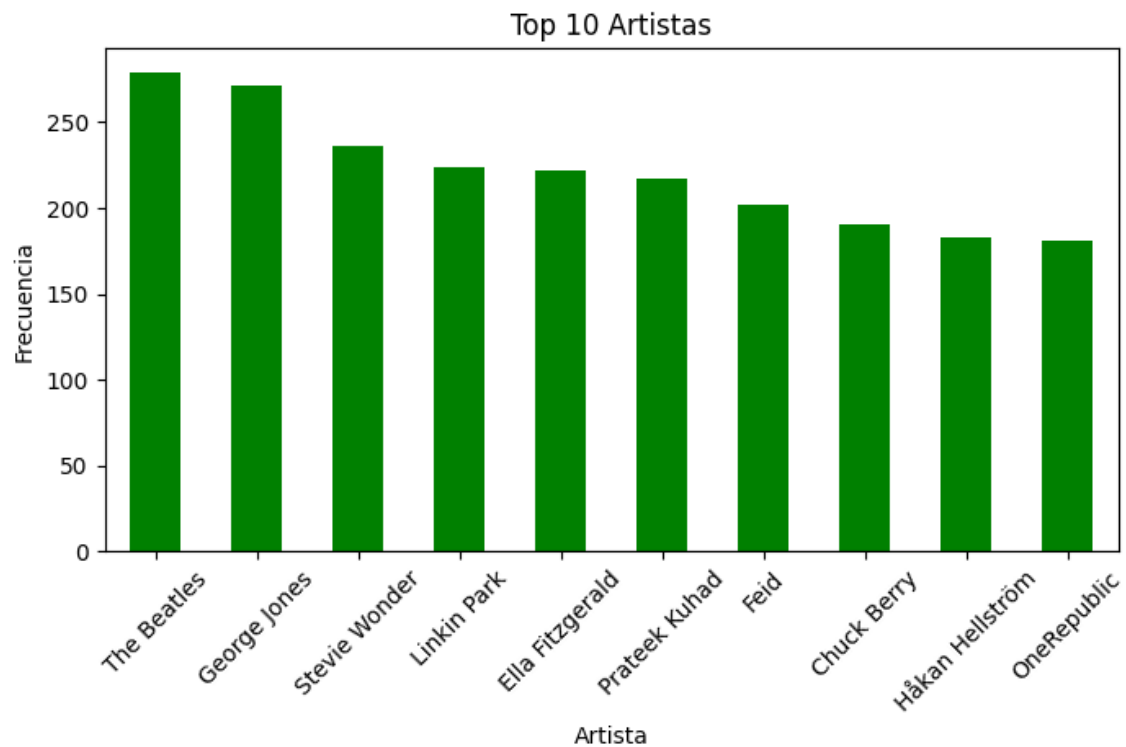
```
rows_with_null_values
```

Executed at 2023.10.14 23:04:39 in 62ms

< 1 row > | 1 rows x 21 columns pd.DataFrame

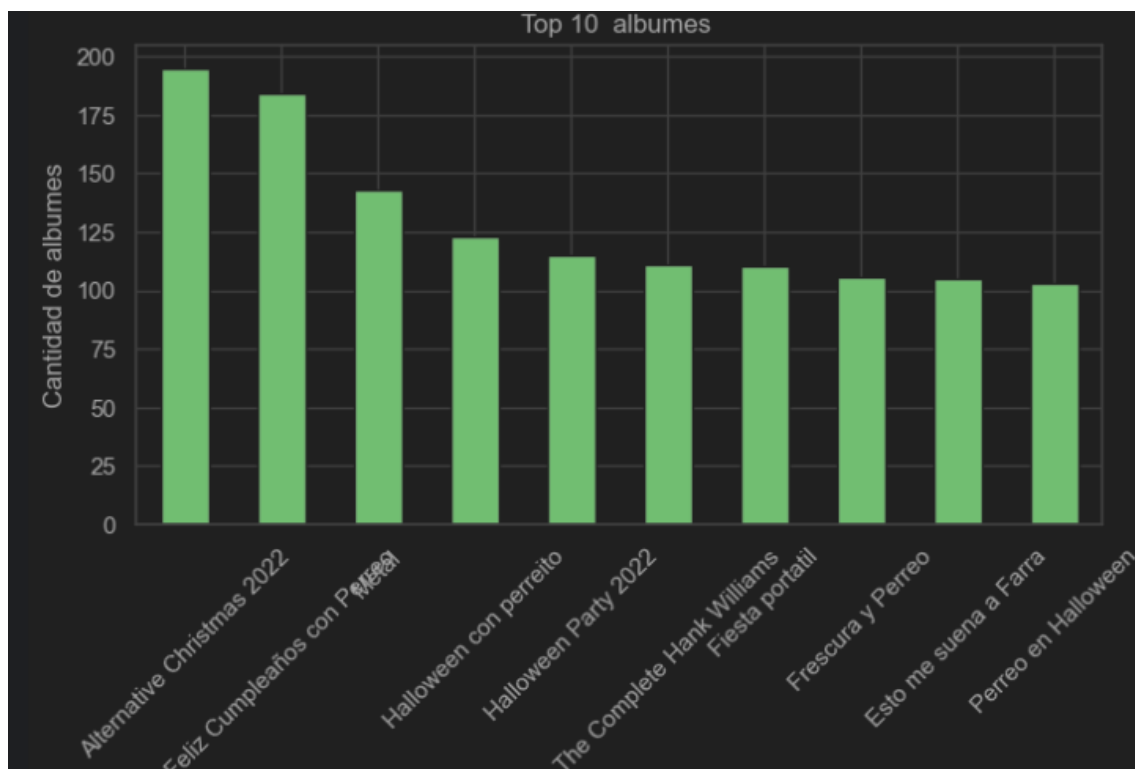
	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceabili
65900	65900	1KR4gIb7nGxHPI3D2ifs59	NaN	NaN	NaN	0	0	False	

Top 10 Artistas Que Mas Aparecen En El Dataset



Podemos Observar que The Beatles son Los que mas aparecen en el conjunto de Datos, tambien se puede ver que la mayoría de los artistas que con mas frecuencia aparecen en el dataset, son cantantes en Ingles.

Podemos ver los 10 mejores albums

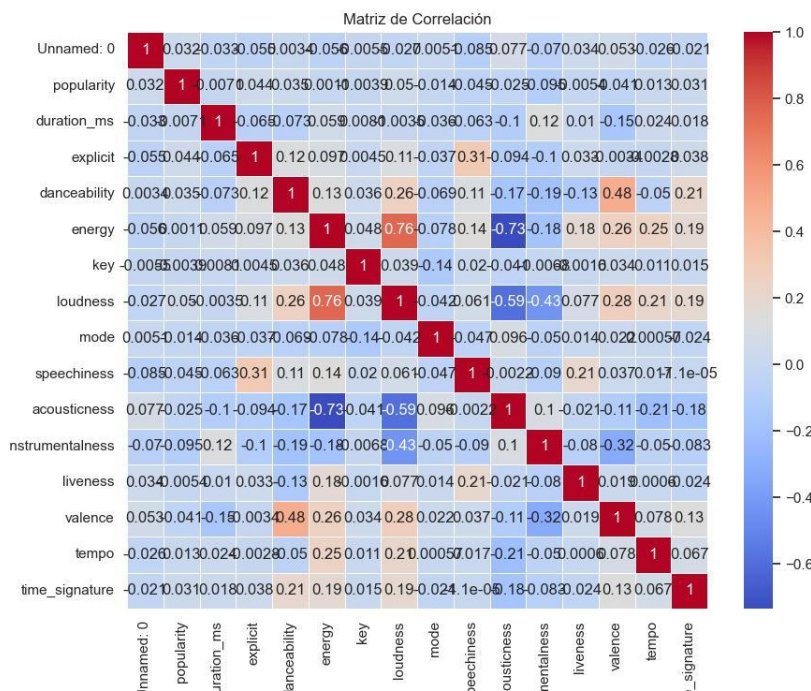


Podemos ver que los álbumes que tienen mas presencia son: Alternative Christmas 2022, Feliz Cumpleaños con Perreo y Metal.

Corplot

Los que tienen colores mas pronunciados a el color azul es por que tienen correlacion positiva, es decir que mientras el uno sube el otro tiende a subir tambien.

el color roja es por que tienen correlacion negativa , es decir que mientras el uno sube el otro tiende a bajar.



Loudness y energy tiene una gran correlacion, acousticness y energy tienen una correlacion negativa, es decir que mientras el uno sube el otro tiende a bajar.

Haciendo un analisis Descriptivos de las Variables:

Tipo Categoricas

El artista que mas se repite son The Beatles y el album que mas se encuentra en el dataset es Alternative Christmas 2022 con 195 apariciones.

df.describe(include=object)
Executed at 2023.10.14 22:49:58 in 188ms

	track_id	artists	album_name	track_name	track_genre
count	114000	113999	113999	113999	114000
unique	89741	31437	46589	73608	114
top	6S3JLDAGk3uu3NtZbPnuhS	The Beatles	Alternative Christmas 2022	Run Rudolph Run	acoustic
freq	9	279	195	151	1000

EDA the_grammys_awards.csv

Hacemos esta llamada a la funcion, para probar que si se subieron los datos en la base de datos

```
def read_db():
    # Hacer la consulta SQL a Grammys
    query = "SELECT * FROM grammys"
    # Crear el engine para conectarse a la base de datos
    engine = creating_engine()
    grammys_df = pd.read_sql(query, con=engine)
    # Cerramos la conexion a la db
    disposing_engine(engine)
    print(grammys_df)

    print(grammys_df)
    logging.info("database read succesfully")

    return grammys_df.to_json(orient='records')

read_db()
```

Efectivamente vemos que la funcion lee los datos

```
PS C:\Users\Guatavo\Documents\5toSemestre\airflow\dags> python etl.py
Engine cerrado
```

	year				title	...	img	winner
0	2019	62nd Annual GRAMMY Awards	(2019)	...	https://www.grammy.com/sites/com/files/styles/...		True	
1	2019	62nd Annual GRAMMY Awards	(2019)	...	https://www.grammy.com/sites/com/files/styles/...		True	
2	2019	62nd Annual GRAMMY Awards	(2019)	...	https://www.grammy.com/sites/com/files/styles/...		True	
3	2019	62nd Annual GRAMMY Awards	(2019)	...	https://www.grammy.com/sites/com/files/styles/...		True	
4	2019	62nd Annual GRAMMY Awards	(2019)	...	https://www.grammy.com/sites/com/files/styles/...		True	
...
4805	1958	1st Annual GRAMMY Awards	(1958)	...			None	True
4806	1958	1st Annual GRAMMY Awards	(1958)	...	https://www.grammy.com/sites/com/files/styles/...		True	
4807	1958	1st Annual GRAMMY Awards	(1958)	...			None	True
4808	1958	1st Annual GRAMMY Awards	(1958)	...			None	True
4809	1958	1st Annual GRAMMY Awards	(1958)	...			None	True

[4810 rows x 10 columns]

El conjunto de datos se distribuye en: **4810 Registros o filas y 10 columnas.**

¿Cuántas variables y observaciones tenemos en el conjunto de datos?

```
df.shape
```

Executed at 2023.09.30 14:23:31 in 17ms

(4810, 10)

Observamos que solo hay 10 columnas, las cuales una **columna numerica y una booleana** .

¿Cuántas variables de cada tipo de dato tenemos en el conjunto de datos?

Add Code Cell Add Markdown Cell

```
(
    df
    .dtypes
    .value_counts()
)
```

Executed at 2023.09.30 14:23:12 in 41ms

```
object      8
int64       1
bool        1
dtype: int64
```

Tenemos 4 columnas con datos nulos.

- nominee:
- artist
- workerr
- img

< < 10 rows > > Length: 10, dtype: bool pd.Series		
	<unnamed>	
updated_at	False	
category	False	
nominee	• True	
artist	• True	
workers	• True	
img	• True	
winner	False	

Cuantos nulos hay por cada variable:

< < 10 rows > > Length: 10, dtype: int64 pd.Series		
	<unnamed>	
workers	2190	
artist	1840	
img	1367	
nominee	6	
year	0	
title	0	
published_at	0	

Para un total de 5403

Borramos los datos duplicados

¿Existen valores nulos explicitos en el conjunto de datos?

```
1 ( df
2   .isnull()
3   .any()
4 )
```

Executed at 2023.10.14 22:19:10 in 144ms

|< < 1-10 > >| Length: 21, dtype: bool [pd.Series ↗](#)

	<unnamed>
Unnamed: 0	False
track_id	False
artists	• True
album_name	• True
track_name	• True
popularity	False
duration_ms	False

Cuantos nulos hay por cada variable:

	<unnamed>
workers	2190
artist	1840
img	1367
nominee	6
year	0
title	0
published_at	0

Para un total de 5403

Borramos los datos duplicados

Cuantos Duplicados Tenemos?

```
1 duplicados = df.duplicated()
2 duplicados.sum()
```

Executed at 2023.10.14 18:29:09 in 285ms

0

Haciendo un analisis Descriptivos de las Variables:

Tipo Categoricas

La gran mayoría de las Canciones fueron publicadas en 2017-11-28T00:03:45-08:00, lo cual es muy raro.

```
df.describe(include=object)
```

Executed at 2023.09.30 14:34:18 in 169ms

	title	published_at	updated_at	category	nominee
count	4810	4810	4810	4810	4804
unique	62	4	10	638	4131
top	62nd Annual GRAMMY Awards (2019)	2017-11-28T00:03:45-08:00	2019-09-10T01:08:19-07:00	Song Of The Year	Bridge Over Troubled Wate
freq	433	4205	778	70	7

La Categoría Mas Nominada durante todos los años ha sido **Song Of The Year**, Tambien podemos evidenciar que la cancion mas nominadaes **Bridge Over Troubled Wate**, con 7 nominaciones.

Tipo Numericas

Solo las numéricas

```
df.describe(include=[np.number])
```

Executed at 2023.09.30 14:27:50 in 64ms

	year
count	4810.000000
mean	1995.566944
std	17.149720
min	1958.000000
25%	1983.000000
50%	1998.000000
75%	2010.000000
max	2019.000000

Podemos ver que los años, van desde **1958 a 2019**.

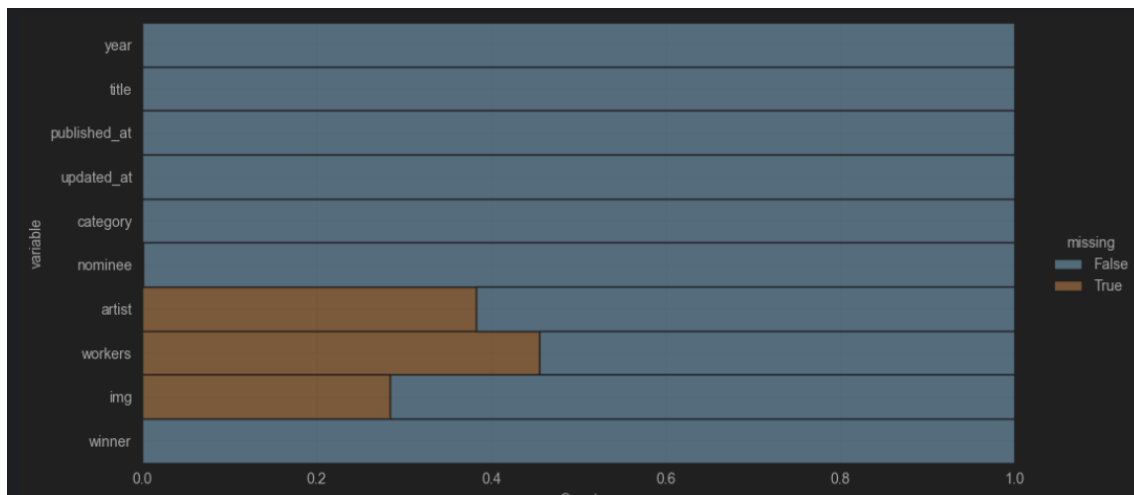
Como podemos observar, La Columna winner solo tiene un valor y es True, entonces lo mas probable es que no todos sea ganadores, si que no que todos fueron nominados.

```
df.describe(include=bool)
```

Executed at 2023.09.30 14:34:29 in 158ms

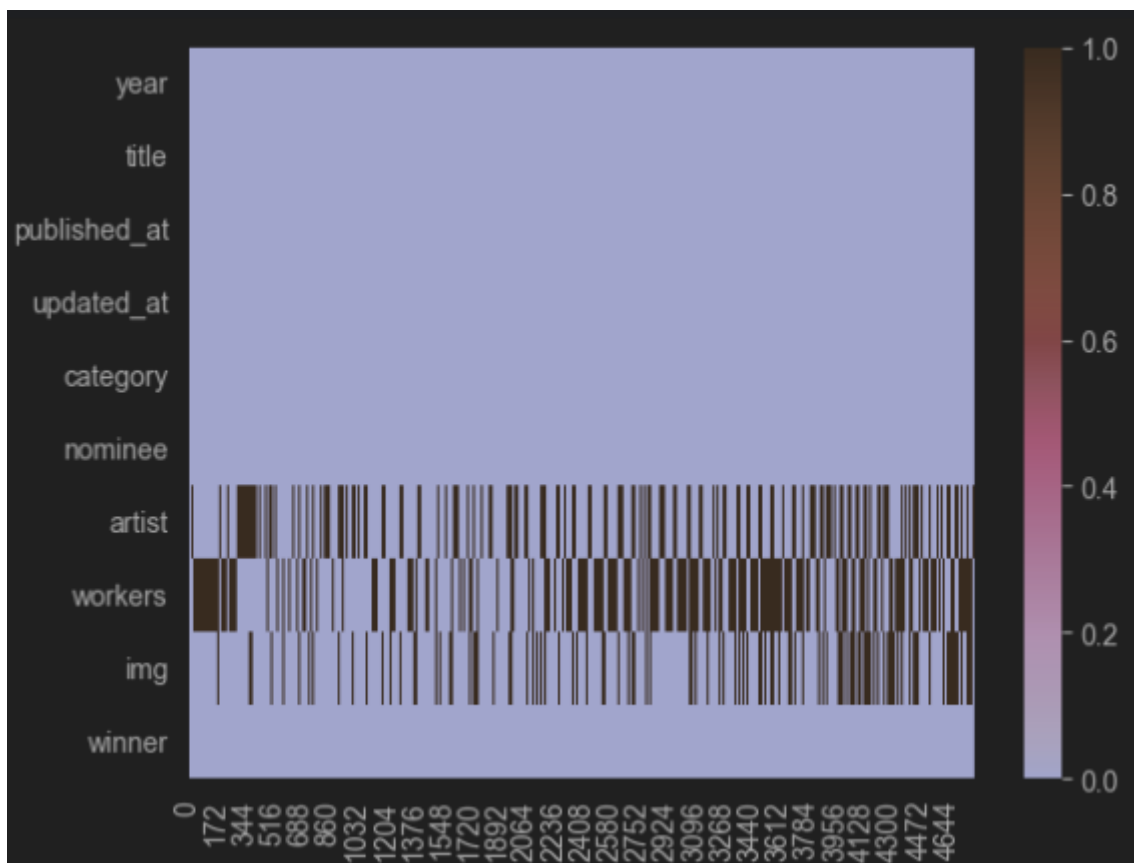
	winner
count	4810
unique	1
top	True
freq	4810

Hacemos un analisis de las proporciones de los nulos en las columnas.



Este grafico nos muestra la proporcion de nulos en cada una de las columnas, donde podemos ver de que la columna img tiene casi un 30 % de los datos nulos, y la columna artist tiene casi un 40% de los valores nulos, la columna workers mas de un 40%.

Ademas del analisis de las columnas, es importante ver como se distribuyen los nulos en las filas, para ver si hay nulos que tienen todos los registros nulos o su gran mayoria. Este grafico se interpreta de manera vertical donde cada raya, representa un registro, y depende lo larga que sea o por donde pase esa raya significa, que es donde ese registro tiene nulas esas columnas.



Podemos analizar que la gran mayoría de los registros se distribuyen solamente en 3 columnas:img,workers y artist.

Pero podremos borrar todos los nulos?

¿Cuántas observaciones perdemos si eliminamos los datos faltantes?

```
df_without_nulls = (
    df
    .dropna()
)

df_without_nulls
```

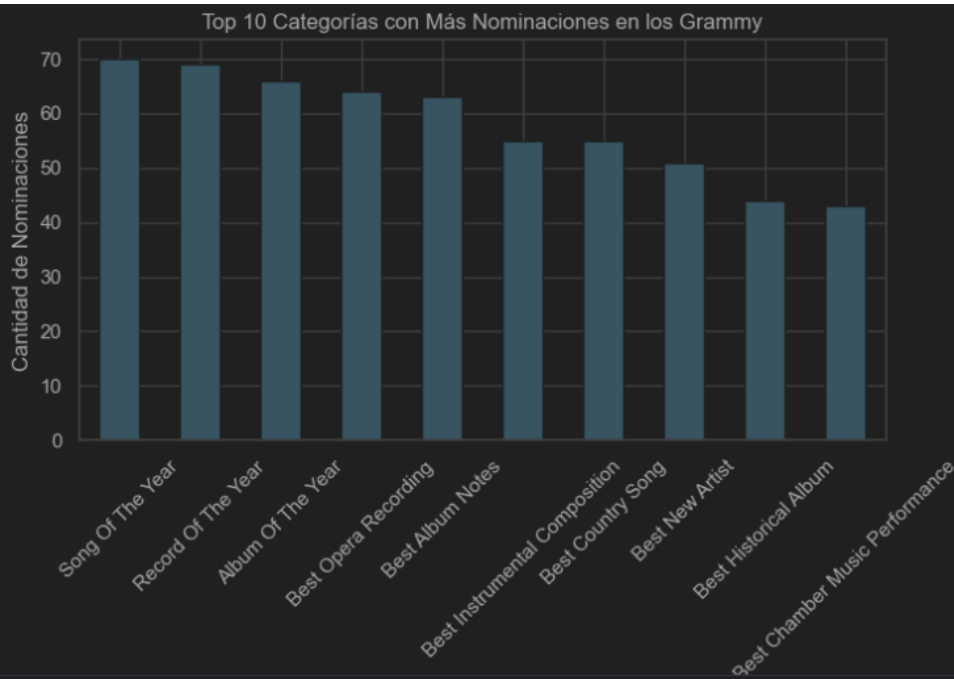
Executed at 2023.09.30 14:53:48 in 170ms

4449	1966	GRAMMY Awards (1966)	28T00:03:45-08:00	10T01:07:37-07:00	Record Of The Year	In The Night	Frank Sinatra	Bowen, producer	https://www.grammy.com/sites/com/files/sty	14	4	627
4450	1966	9th Annual GRAMMY Awards (1966)	2017-11-28T00:03:45-08:00	2019-09-10T01:07:37-07:00	Album Of The Year	A Man And His Music	Frank Sinatra	Sonny Burke, producer	https://www.grammy.com/sites/com/files/styles/...			True
4492	1965	8th Annual GRAMMY Awards (1965)	2017-11-28T00:03:45-08:00	2019-09-10T01:06:59-07:00	Record Of The Year	A Taste Of Honey	Herb Alpert And The Tijuana Brass	Herb Alpert & Jerry Moss, producers	https://www.grammy.com/sites/com/files/styles/...			True
4493	1965	8th Annual GRAMMY Awards (1965)	2017-11-28T00:03:45-08:00	2019-09-10T01:06:59-07:00	Album Of The Year	September Of My Years	Frank Sinatra	Sonny Burke, producer	https://www.grammy.com/sites/com/files/styles/...			True
4629	1962	5th Annual GRAMMY Awards (1962)	2017-11-28T00:03:45-08:00	2019-09-10T01:09:02-07:00	Album Of The Year (Other Than Classical)	The First Family	Vaughn Meader	Bob Booker & Earle Doud, producers	https://www.grammy.com/sites/com/files/styles/...			True

834 rows x 10 columns

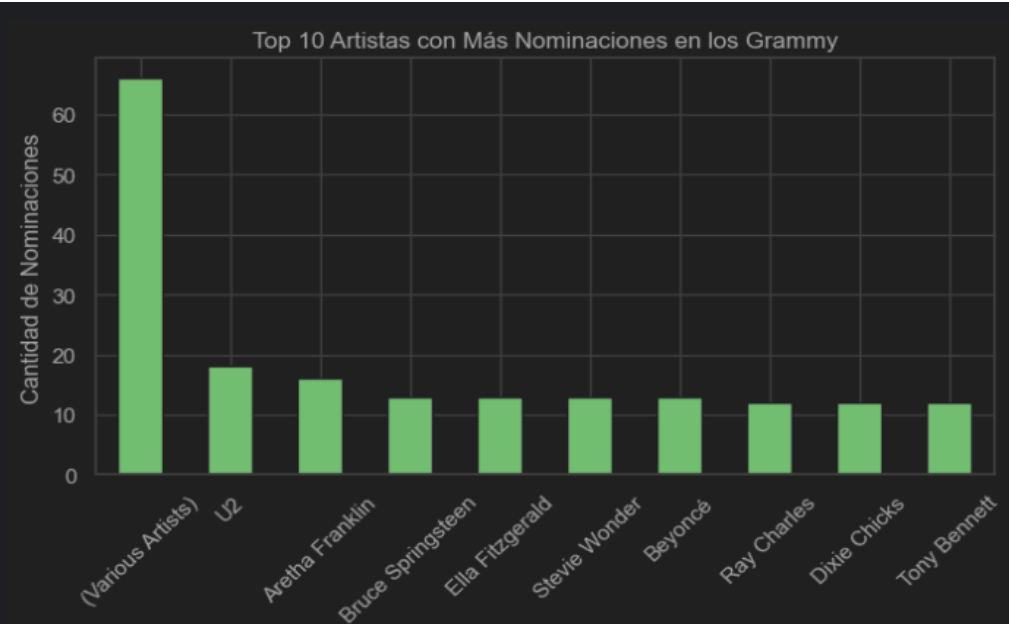
Miramos que quedaríamos con menos del 25% de nuestra informacion inicial, lo cual no es nada satisfactorio, entonces indagamos mas en las columnas que tienen nulos.

Top 10 Categorías con Mas Nominaciones:



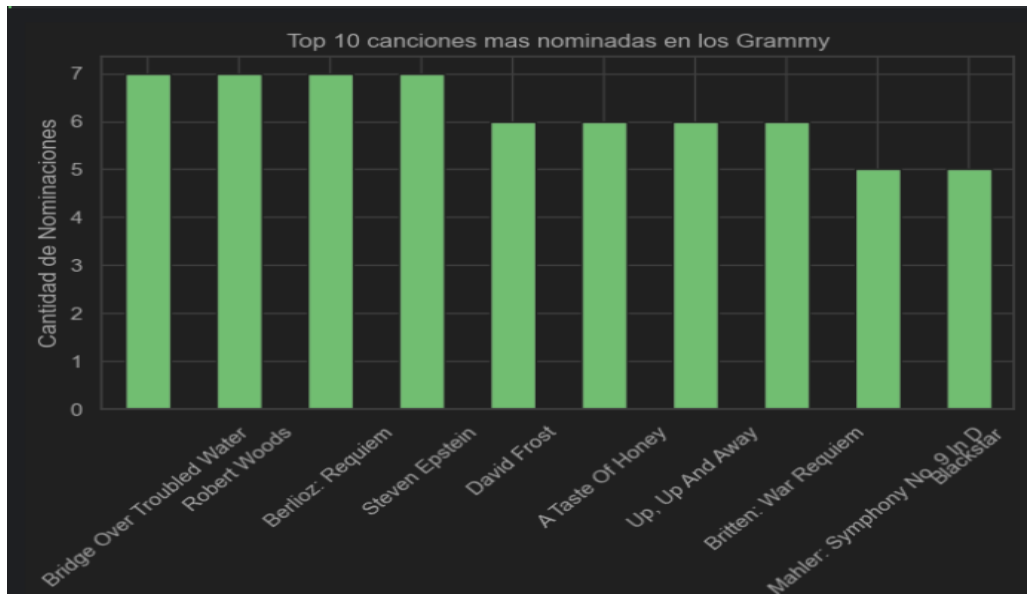
Podemos observar que las categorías Song, Record y Album Of The Year, son las categorías que mas se repiten, con una frecuencia de mas de 65 veces cada una, esto nos da a entender que todos los años almenos una vez se entrega este premio

Top 10 Artistas con Mas Nominaciones :



Se evidencia la existencia de un dato atipico , **Various Artists**. El cual no nos interesará analizar y será eliminado, ya que queremos analizar artistas en particular. Además podemos ver que los artistas con mas nominaciones (sacando el dato atipico) son **U2 ,Aretha Fraklin y Bruce Springsteen**.

Top 10 Artistas con Mas Nominaciones :



Se puede mirar un empate en el primer lugar ya que hay 4 canciones que tienen 7 nominaciones y son:**Bridge Over Troubled Water, Robert Woods, Beriloz Requiem y Steven Epstein**.

Transformaciones Dataset Grammys:

Todas las transformaciones, tanto del dataset de Grammys como de Spotify, estan en el archivo `transfom.py`

Funciones: **no_needed_columns** , **borrar_duplicados**, **drop_null_rows**,
Transform_winner_column

no_needed_columns

```
def no_needed_columns(df):  
    df.drop(['img', 'title', 'published_at', 'updated_at', 'workers'], axis=1, inplace=True)  
    return df
```

se eliminan `img`, `workers`, `updated_at`, `published_at` porque no nos aportan informacion relevante para nuestro analisis. `title` se eliminan porque se puede inferir de que año son los premios de los grammys con la columna `year`

Transform_winner_column

```
def Transform_winner_column(df):  
    df['winner'] = df['winner'].map({True: 1, False: 0})  
    return df
```

Entendiendo que `True` significa que ganaron los premios, pero al 100 % de la columna tiene `True` , nos hace inferir de que lo mas probable es que todos los que estan en el dataset ganaron, entonces cambiamos el valor `True` por `1`.

drop_duplicates

Boramos los datos duplicados.

```
#duplicados  
def borrar_duplicados(df):  
    df.drop_duplicates(keep='first',inplace=True)  
    return df
```

drop_null_rows

Verificamos si existen mas registros que empiecen con nombre `REMIXER` en la columna `category`

para validar si hay registros con el mismo nombre en la columna "`category`" y asi rellenarlo, de no ser asi seran borrados


```
new_df = df[df["category"].str.startswith("REMIXER")]
new_df
```

Executed at 2023.09.30 16:36:20 in 176ms

#	year	title	published_at	updated_at	category
2261	2000	43RD ANNUAL GRAMMY AWARDS (2000)	2017-11-28T00:03:45-08:00	2019-09-10T01:11:09-07:00	REMIXER OF THE YEAR, NON-CLASSICAL
2359	1999	42ND ANNUAL GRAMMY AWARDS (1999)	2017-11-28T00:03:45-08:00	2019-09-10T01:09:02-07:00	REMIXER OF THE YEAR, NON-CLASSICAL
2454	1998	41ST ANNUAL GRAMMY AWARDS (1998)	2017-11-28T00:03:45-08:00	2019-09-10T01:08:19-07:00	REMIXER OF THE YEAR, NON-CLASSICAL
2547	1997	40TH ANNUAL GRAMMY AWARDS (1997)	2017-11-28T00:03:45-08:00	2019-09-10T01:07:37-07:00	REMIXER OF THE YEAR, NON-CLASSICAL

Como vemos no hay mas registros, entonces por eso son eliminados

Verificamos si existen mas registros que empiecen con nombre BEST NEW COUNTRY & en la columna category para validar si rellenamos o borramos estos registros

```
new_df = df[df["category"].str.startswith("BEST NEW COUNTRY &")]
new_df
```

Executed at 2023.09.30 16:41:07 in 246ms

#	year	title	published_at	updated_at	category
4525	1965	8TH ANNUAL GRAMMY AWARDS (1965)	2017-11-28T00:03:45-08:00	2019-09-10T01:06:59-07:00	BEST NEW COUNTRY & WESTERN ARTIST
4573	1964	7TH ANNUAL GRAMMY AWARDS (1964)	2017-11-28T00:03:45-08:00	2019-09-10T01:06:11-07:00	BEST NEW COUNTRY & WESTERN ARTIST OF

```
def drop_null_rows(df):
    df.drop([2261,2359,2454,2547,4525,4573], axis=0, inplace=True)
    return df
```

Eliminamos los registros ya que tienen columnas importantes para nuestro analisis vacias y ademas no se vio la forma de rellenarlos

def change_categories:

En esta funcion agrupamos las categorias , ya que existen mas de 600 categorias, las reducimos a 41, las cuales pueden ser analizadas mucho mas sencillo.

1.(explicando con ejemplo)

En el Caso de las categorias:

'Best Song Written For Visual Media',
 'Best Compilation Soundtrack For Visual Media',
 'Best Score Soundtrack For Visual Media',
 'Best Music Video',
 'Best Music Film'

Seran cambiadas su nombre por

Soundtracks/Music Videos y asi sucesivamente con todos.(Puede revisar el archivo **categories1.pdf** para mirar todas las agrupaciones que se hicieron en esta columna)

```
categories = {
    'Soundtracks/Music Video': [
        'Best Song Written For Visual Media',
        'Best Compilation Soundtrack For Visual Media',
        'Best Score Soundtrack For Visual Media',
        'Best Music Video',
        'Best Music Film'
    ],
    'Production/Engineering': [
        'Best Instrumental Arrangement',
        'Best Arrangement, Instrumental or A Cappella',
        'Best Arrangement, Instruments and Vocals',
        'Best Recording Package',
        'Best Boxed Or Special Limited Edition Package',
    ]
}
```

Esta función **etiquetar_categoria** que está dentro de la función **def change_categories**, lo que nos ayuda es a recorrer la variable **categories** y con la función **apply**, agrupamos cada una de las variables explicadas en el **punto 1.(explicando con ejemplo)**

```
},}
def etiquetar_categoria(category):
    for clave, palabras_clave in categories.items():
        for palabra_clave in palabras_clave:
            if palabra_clave in category:
                return clave
    return category
df['category'] = df['category'].apply(etiquetar_categoria)
return df
```

Verificamos que después de usar la función se reduce a 41 los diferentes datos de la columna

```
valores_unicos_category1 = df["category"].nunique()
valores_unicos_category1
```

Executed at 2023.10.14 22:17:05 in 31ms

627

```
valores_unicos_category1 = df["category"].nunique()
valores_unicos_category1
```

Executed at 2023.10.14 22:11:13 in 62ms

41

Transformaciones, Dataset Spotify:

Funciones: **agregando_la_duracion_en_minutos**, **drop_unnamed_column**, **borrar_duplicados**, **borrar_nulos_spotify**, **categorizar_duracion**, **categorizar_popularidad**,

1. **agregando_la_duracion_en_minutos(df)**: Esta función toma un DataFrame **df** y agrega una nueva columna llamada 'duration_min', que representa la duración en minutos en lugar de milisegundos. Los valores se redondean a 2 decimales.

```
#crea una nueva columna pasando los ms a minutos
def agregando_la_duracion_en_minutos(df):
    df['duration_min'] = (df['duration_ms'] / 60000).round(2)
    #df.head(2)
    return df
```

2. **drop_unnamed_column(df)**: Elimina la columna 'Unnamed: 0' del DataFrame **df**. La borramos ya que esta columna no nos aporta valor

```
def drop_unnamed_column(df):
    df.drop(columns=["Unnamed: 0"], axis=1, inplace=True)
    return df
```

3. **borrar_duplicados(df)**: Elimina filas duplicadas del DataFrame **df**.

```
def borrar_duplicados(df):
    df.drop_duplicates(keep='first',inplace=True)
    return df
```

4. **borrar_nulos_spotify(df)**: Elimina un registro específico (fila 65900) del DataFrame **df**. Ya que este registro tiene nulos en estos 3 columnas: album_name,artist,track_name los cuales son altamente importantes para el analisis.

```
#borra el registro nulo
def borrar_nulos_spotify(df):
    df.drop([65900], axis=0, inplace=True)
    return df
```

5. **categorizar_duracion(categorizar_duracion)**: Esta función define una categoría para la duración en minutos. Si el valor de duración está por debajo de 3, se etiqueta como 'Poca_Duración', si está entre 3 y 4 se etiqueta como 'Duración_Media', y de lo contrario se etiqueta como 'Mucha_Duración'.

```
def categorizar_duracion(categorizar_duracion):
    if categorizar_duracion <= 3:
        return 'Poca_Duración'
    elif 3.0 < categorizar_duracion <= 4.0 :
        return 'Duración_Media'
    else:
        return 'Mucha_Duración'
    # Aplica la función para crear la nueva columna 'duration_category'
```

6. **categorizar_popularidad(popularidad):** Similar a la función anterior, esta función asigna una categoría a la popularidad de una canción. Si la popularidad es menor o igual a 30, se etiqueta como 'Popularidad Baja', si está entre 30 y 60 se etiqueta como 'Popularidad Media', y de lo contrario se etiqueta como 'Popularidad Alta'.

```
# Define una función que asigna la categoría según la popularidad
def categorizar_popularidad(popularidad):
    if popularidad <= 30:
        return 'Popularidad Baja'
    elif 30 < popularidad <= 60:
        return 'Popularidad Media'
    else:
        return 'Popularidad Alta'
    # Aplica la función para crear la nueva columna 'popularity_category'
```

7. **crear_columnas_categoricas(df):** Esta función crea columnas categóricas basadas en umbrales para las características 'danceability', 'energy' y 'valence'. Utiliza la función **pd.cut()** para dividir los valores en categorías de 'Baja', 'Media' y 'Alta'. Para asignarcelos a 'danceability' y 'energy' dependiendo si están entre 0 y 0.3, si están entre 0.3 y 0.7, si están entre 0.7 y 1.0. También crea una columna 'valence_category' con etiquetas 'Negativa', 'Neutral' y 'Positiva'. Asignándole los valores, igualmente usando la misma escala.

Luego el **df['popularity_category']** y **df['duration_category']** donde se llaman las funciones **categorizar_duracion** y **categorizar_popularidad** las cuales retornan si la popularidad y la duración es alta, media o baja respectivamente.

```
def crear_columnas_categoricas(df):
    # Define los umbrales para las categorías
    umbrales_danceability = [0.0, 0.3, 0.7, 1.0]
    umbrales_energy = [0.0, 0.3, 0.7, 1.0]
    umbrales_valence = [0.0, 0.3, 0.7, 1.0]
    # Define los nombres de las categorías
    categorias = ['Baja', 'Media', 'Alta']
    categorias_valence = ['Negativa', 'Neutral', 'Positiva']

    # Utiliza pd.cut() para crear la nueva columna categórica
    df['danceability_category'] = pd.cut(df['danceability'], bins=umbrales_danceability, labels=categorias)
    df['energy_category'] = pd.cut(df['energy'], bins=umbrales_energy, labels=categorias)
    df['valence_category'] = pd.cut(df['valence'], bins=umbrales_valence, labels=categorias_valence)
    df['popularity_category'] = df['popularity'].apply(categorizar_popularidad)
    df['duration_category'] = df['duration_min'].apply(categorizar_duracion)
    return df
```

Documentación- bd_Connection.py

1. Introducción El módulo "bd_Connection.py" proporciona funcionalidad para gestionar la conexión a una base de datos PostgreSQL. En este documento, se describen las funciones y variables clave en este módulo.

2. Importaciones

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import json
```

3. Variables

2.1 db_params (dict) Un diccionario que contiene los parámetros de la base de datos, incluyendo usuario, contraseña, host, puerto y nombre de la base de datos.

2.2 db_url (str) La URL de conexión generada a partir de los parámetros de la base de datos.

3. Funciones

3.1 creating_engine() Esta función crea un motor (engine) de SQLAlchemy para la base de datos y lo retorna.

```
# Función para crear el motor (engine)
def creating_engine():
    engine = create_engine(db_url)
    return engine
```

3.2 creating_session(engine) Crea una sesión de SQLAlchemy a partir de un motor dado y la retorna.

```
# Función para crear las sesiones
def creating_session(engine):
    Session = sessionmaker(bind=engine)
    session = Session()
    return session
```

3.3 closing_session(session) Cierra la sesión de SQLAlchemy especificada.

```
# Función para cerrar la sesión
def closing_session(session):
    session.close()
```

3.4 disposing_engine(engine) Cierra y elimina el motor de SQLAlchemy especificado.

```
# Función para cerrar el motor (engine)
def disposing_engine(engine):
    engine.dispose()
    print("Engine cerrado")
```

4. Uso

Para utilizar las funciones proporcionadas en este módulo, sigue estos pasos:

4.1 Importa las funciones necesarias:

```
from bd_Connection import creating_engine, creating_session, closing_session,  
disposing_engine
```

4.2 Crea un motor y una sesión:

```
engine = creating_engine() session = creating_session(engine)
```

4.3 Realiza las operaciones necesarias en la base de datos utilizando la sesión.

4.4 Cierra la sesión cuando hayas terminado:

```
closing_session(session)
```

4.5 Finalmente, cierra y elimina el motor cuando ya no sea necesario:

```
disposing_engine(engine)
```

5. Conclusiones

El módulo "bd_Connection.py" facilita la conexión y gestión de sesiones con una base de datos PostgreSQL. Las funciones proporcionadas permiten una gestión eficiente de las conexiones y garantizan su cierre adecuado.

Espero que esta estructura te sea útil para crear la documentación en Word. Puedes copiar y pegar el contenido en un documento de Word y darle formato según tus preferencias.

Documentación- load_To_Drive.py

1. Introducción

El archivo "load_To_Drive.py" contiene funciones que permiten autenticarse en Google Drive y cargar archivos en una ubicación específica de la unidad de Google Drive. Este documento describe las funciones clave y su uso.

2. Variables

2.1 directorio_credenciales (str)

- Ruta al archivo de credenciales JSON necesario para la autenticación en Google Drive.

3. Funciones

3.1 login()

- Esta función inicia sesión en Google Drive. Realiza los siguientes pasos:
 - Configura las credenciales del cliente utilizando el archivo especificado en **directorio_credenciales**.
 - Intenta cargar credenciales previas si están disponibles.

- Realiza la autenticación a través de un servidor web local si las credenciales no están disponibles o han expirado.
- Refresca las credenciales si están vencidas.
- Autoriza la aplicación para acceder a Google Drive.
- Guarda las credenciales para futuras sesiones.
- Retorna una instancia de GoogleDrive autorizada.

3.2 `subir_archivo(ruta_archivo, id_folder)`

- Esta función carga un archivo local en Google Drive en la ubicación especificada por **id_folder**.
- Parámetros:
 - **ruta_archivo** (str): La ruta local al archivo que se va a cargar.
 - **id_folder** (str): El ID de la carpeta en Google Drive donde se desea cargar el archivo.
- La función utiliza la instancia de GoogleDrive obtenida a través de la función **login()** para cargar el archivo en Google Drive.

Documentación- etl_dag.py

1. Introducción

El archivo "etl_dag.py" define un DAG (Grafo Acíclico Dirigido) de Apache Airflow para orquestrar un proceso de ETL (Extract, Transform, Load) que se encarga de leer datos desde una base de datos, transformarlos y cargarlos en Google Drive. Este documento describe las tareas y la configuración del DAG.

2. Configuración del DAG

2.1 `default_args` (dict)

- Un diccionario que contiene argumentos predeterminados para las tareas del DAG. Incluye información sobre el propietario, la frecuencia de ejecución, notificaciones por correo electrónico y reintento en caso de fallos.

2.2 Creación del DAG

- Se crea un objeto DAG llamado "etl_dag" con la configuración proporcionada en **default_args**. La descripción y la frecuencia de ejecución del DAG se pueden personalizar.

3. Tareas del DAG

El DAG consta de varias tareas que representan diferentes etapas del proceso ETL:

3.1 `read_db`

- Tarea que ejecuta la función **read_db** para leer datos desde la base de datos los datos de grammys.

3.2 **grammys_transform_db**

- Tarea que ejecuta la función **grammys_transform_db** para realizar transformaciones específicas en los datos obtenidos de la base de datos.

3.3 **read_csv**

- Tarea que ejecuta la función **read_csv** para leer datos desde un el CSV de los grammys.

3.4 **transform_spotify**

- Tarea que ejecuta la función **transform_spotify** para realizar transformaciones en los datos de Spotify.

3.5 **merge**

- Tarea que ejecuta la función **merge** para combinar los datos transformados(spotify y grammys).

3.6 **load**

- Tarea que ejecuta la función **load** para cargar los datos combinados la base de datos.

3.7 **load_drive**

- Tarea que ejecuta la función **load_drive** para cargar los datos en Google Drive.

4. Orquestación de Tareas

Las tareas del DAG están orquestadas de la siguiente manera:

- **read_db** y **grammys_transform_db** se ejecutan en paralelo.
- **read_csv** y **transform_spotify** se ejecutan en paralelo.
- Después de que las tareas anteriores estén completas, **merge** combina los datos.
- Finalmente, los datos se cargan en Google Drive mediante la secuencia **merge -> load -> load_drive**.

5. Conclusiones

El archivo "etl_dag.py" define un DAG de Apache Airflow que automatiza el proceso ETL, permitiendo la lectura, transformación y carga de datos desde diversas fuentes hacia Google Drive. La orquestación de tareas permite una gestión eficiente de este flujo de trabajo.

Documentación- etl.py

1. Introducción

El archivo "etl.py" contiene funciones que forman parte del proceso ETL (Extract, Transform, Load) para la manipulación de datos. Estas funciones leen datos desde diferentes fuentes, los transforman y finalmente los cargan en una base de datos y en Google Drive. En este documento, describiremos cada función y su objetivo.

2. Funciones

2.1 read_csv()

- Esta función lee datos desde un archivo CSV llamado "spotify_dataset.csv" y devuelve el DataFrame resultante.

2.2 transform_spotify(**kwargs)

- Esta función realiza varias transformaciones en el DataFrame de Spotify. Utiliza las funciones importadas de "transform.py" para eliminar columnas no necesarias, calcular la duración en minutos, categorizar datos y eliminar duplicados y nulos.

2.3 read_db()

- Esta función realiza una consulta SQL en una base de datos llamada "grammys" y devuelve el DataFrame resultante.

2.4 grammys_transform_db(**kwargs)

- Esta función transforma el DataFrame obtenido de la base de datos "grammys". Realiza transformaciones como la eliminación de columnas no necesarias, la conversión de la columna 'winner', la eliminación de duplicados y la eliminación de registros nulos.

2.5 merge(**kwargs)

- Esta función combina los DataFrames de Spotify y Grammy en función de la columna 'track_name'. El resultado es un DataFrame combinado.

2.6 load(**kwargs)

- Esta función carga el DataFrame combinado en una base de datos utilizando SQLAlchemy. También guarda una copia del DataFrame en un archivo CSV llamado "merged_data.csv".

2.7 load_drive(**kwargs)

- Esta función carga el archivo "merged_data.csv" en Google Drive utilizando las funciones definidas en "load_To_Drive.py".

Ejemplo:

Vamos a explicar como se integra el **Etl_dag.py** con el **etl.py** en la funcion merge en esta parte.

```
def merge(**kwargs):
    ti = kwargs["ti"]
    logging.info( f"in the merge function")
    str_data = ti.xcom_pull(task_ids="transform_csv")
    json_data = json.loads(str_data)
    spotify_df = pd.json_normalize(data=json_data)
    #df=read_csv()
    print("adadadad")

    logging.info( f"grammys is entering to the merge function")
    str_data = ti.xcom_pull(task_ids="transform_db")
    json_data = json.loads(str_data)
    grammys_df = pd.json_normalize(data=json_data)
```

ti = kwargs["ti"]

En este fragmento, estamos accediendo a los datos compartidos (XCom) entre tareas en Apache Airflow. "ti" es una instancia de la clase **TaskInstance**, que nos permite acceder a los resultados y atributos de otras tareas en el mismo flujo de trabajo. En este caso, "kwargs" es un diccionario que contiene argumentos adicionales pasados a la función. "ti" se utiliza para acceder a estos argumentos.

pythonCopy code

logging.info(f"in the merge function")

Aquí, estamos registrando un mensaje informativo en los registros (logs) de Apache Airflow. Este mensaje se mostrará en los registros para proporcionar información sobre el progreso del flujo de trabajo. El mensaje indica que estamos en la función de combinación ("merge function").

str_data = ti.xcom_pull(task_ids="transform_csv")

Usamos **ti.xcom_pull()** para extraer datos compartidos de otra tarea. En este caso, estamos extrayendo datos de la tarea con el identificador "transform_csv". Esto nos permite obtener los resultados de la tarea "transform_csv" y utilizarlos en la función actual. Estos resultados se almacenan en "str_data" como una cadena.

json_data = json.loads(str_data)

Luego, convertimos la cadena "str_data" en un objeto JSON utilizando la función **json.loads()**. Esto es necesario porque los datos compartidos suelen ser serializados en formato JSON.

spotify_df = pd.json_normalize(data=json_data)

A continuación, utilizamos la biblioteca Pandas para normalizar los datos JSON y crear un DataFrame llamado "spotify_df". Esto nos permite trabajar con los datos en un formato tabular que se puede utilizar en las operaciones posteriores.

En resumen, esta sección del código se encarga de obtener los datos de dos tareas anteriores ("transform_csv" y "transform_db") utilizando el mecanismo de datos compartidos de Apache Airflow. Luego, convierte estos datos en DataFrames de Pandas para que puedan ser combinados en la función de "merge". Estos pasos son esenciales para realizar la operación de combinación de datos con éxito en el flujo de trabajo ETL.

4. Conclusiones

El archivo "etl.py" define un conjunto de funciones que trabajan en conjunto para realizar un proceso ETL completo. Las funciones leen datos desde diferentes fuentes, los transforman de manera efectiva y los cargan en un destino, lo que facilita la manipulación de datos en proyectos de análisis y almacenamiento.

EVIDENCIAS

Load :

Podemos ver de que el merge fue subido a la base de datos:

Esta es la funcion merge, donde cargamos los 2 jsons del **transform_db(grammys)** **transform_csv(spotify)**.

```
def merge(**kwargs):
    ti = kwargs["ti"]
    logging.info( f"in the merge function")
    str_data = ti.xcom_pull(task_ids="transform_csv")
    json_data = json.loads(str_data)
    spotify_df = pd.json_normalize(data=json_data)
    #df=read_csv()
    print("adadadad")

    logging.info( f"grammys is entering to the merge function")
    str_data = ti.xcom_pull(task_ids="transform_db")
    json_data = json.loads(str_data)
    grammys_df = pd.json_normalize(data=json_data)

    df = spotify_df.merge(grammys_df, left_on='track_name', right_on='nominee', how='inner')
    print(df)
    print(df.columns)
    logging.info( f"data is done to merge")
    return df.to_json(orient='records')

#df = spotify_df.merge(grammys_df, how='left', left_on='track_name', right_on='nominee')
df = spotify_df.merge(grammys_df, left_on='track_name', right_on='nominee', how='inner')
```

Esta linea en especifico hace la union de los conjunto de datos.

```
def load(**kwargs):
    logging.info("starting load")
    ti = kwargs["ti"]
    str_data = ti.xcom_pull(task_ids="merge")
    json_data = json.loads(str_data)
    df = pd.json_normalize(data=json_data)
    engine = creating_engine()

    df.to_sql('merge', engine, if_exists='replace', index=False)

    #Cerramos la conexion a la db
    disposing_engine(engine)
    df.to_csv("merged_data.csv", index=False)
    logging.info( f"data is done to load")
    return df.to_json(orient='records')
```


















Y aquí subimos el df con el merge a la base tabla merge.

Probamos con el siguiente codigo

```
def read_db():
    # Hacer la consulta SQL a Grammys
    consulta_sql = "SELECT * FROM merge"
    # Crear el engine para conectarse a la base de datos
    engine = creating_engine()
    grammys_df = pd.read_sql(consulta_sql, con=engine)
    #Cerramos la conexion a la db
    disposing_engine(engine)
    print(grammys_df)
read_db()
```

```
[4810 rows x 10 columns]
   track_id      artists  ...  artist winner
0  6Vc5wAMmXdKIAM7wUoEb7N  A Great Big World;Christina Aguilera  ...  A Great Big World With Christina Aguilera  1
1  5TvE3pk05pyFIGdSY9j4DJ  A Great Big World;Christina Aguilera  ...  A Great Big World With Christina Aguilera  1
2  78TKt1SLWK8pZAKKw3MyQL  A Great Big World;Christina Aguilera  ...  A Great Big World With Christina Aguilera  1
3  7xLhousIHDxoGgeJNh04Ye  A Great Big World  ...  A Great Big World With Christina Aguilera  1
4  6kRZf0ZSuoioHtg6gROKDz  A Great Big World  ...  A Great Big World With Christina Aguilera  1
...  ...  ...  ...  ...
3182  61h02ZrTkcoXNt8eBIMJX9  Phil Wickham  ...  Tim Menzies  1
3183  7d1CAK8HdnI6UGHqZkFvJT  for KING & COUNTRY  ...  NaN  1
3184  5B1fUGBm5LF3V9U1AYpLpf  Hillsong Worship  ...  Steven Curtis Chapman  1
3185  41FGj9LMN5xYuGeDcN1fo9  Jadon Lavik  ...  Larry Hart  1
3186  3jId15LURV3fJNmVSnD574  Hillsong Worship  ...  Al Jarreau  1

[3187 rows x 31 columns]
Engine cerrado
```

 _pycache_	10/14/2023 11:03 AM	File folder	
 bd	10/14/2023 1:34 AM	Archivo de origen ...	2 KB
 bd_Connection	10/14/2023 8:21 AM	Archivo de origen ...	2 KB
 bd_query	10/14/2023 12:47 AM	Archivo de origen ...	3 KB
 client_secrets	10/14/2023 12:21 PM	Archivo de origen ...	1 KB
 credentials_module	10/14/2023 12:42 PM	Archivo de origen ...	2 KB
 dag	10/14/2023 11:03 AM	Archivo de origen ...	2 KB
 etl copy	10/14/2023 8:38 AM	Archivo de origen ...	4 KB
 etl	10/14/2023 2:28 PM	Archivo de origen ...	5 KB
 etl_dag	10/14/2023 9:06 AM	Archivo de origen ...	2 KB
 load_To_Drive	10/14/2023 12:42 PM	Archivo de origen ...	2 KB
 merged_data	10/13/2023 3:23 PM	Microsoft Excel C...	859 KB
 quick_Start	10/14/2023 12:14 PM	Archivo de origen ...	1 KB
 settings	10/14/2023 12:25 PM	Archivo de origen ...	1 KB
 spotify_dataset	10/13/2023 11:45 AM	Microsoft Excel C...	19,647 KB
 the_grammy_awards	9/16/2020 9:21 AM	Microsoft Excel C...	1,435 KB
 transform	10/14/2023 10:06 PM	Archivo de origen ...	34 KB

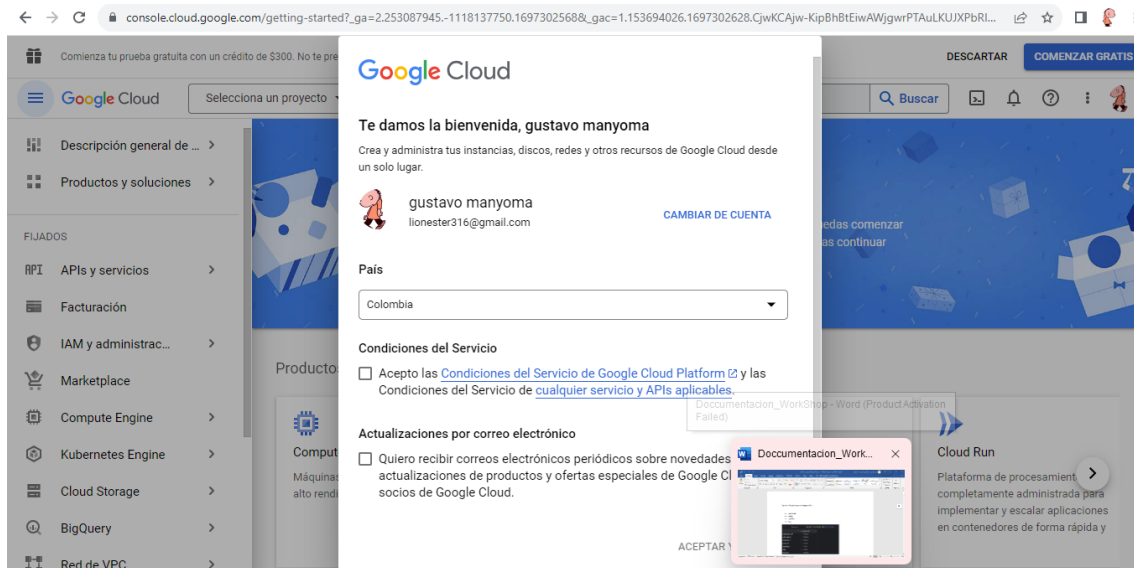
Load_To_Drive

[Link del archivo](#)

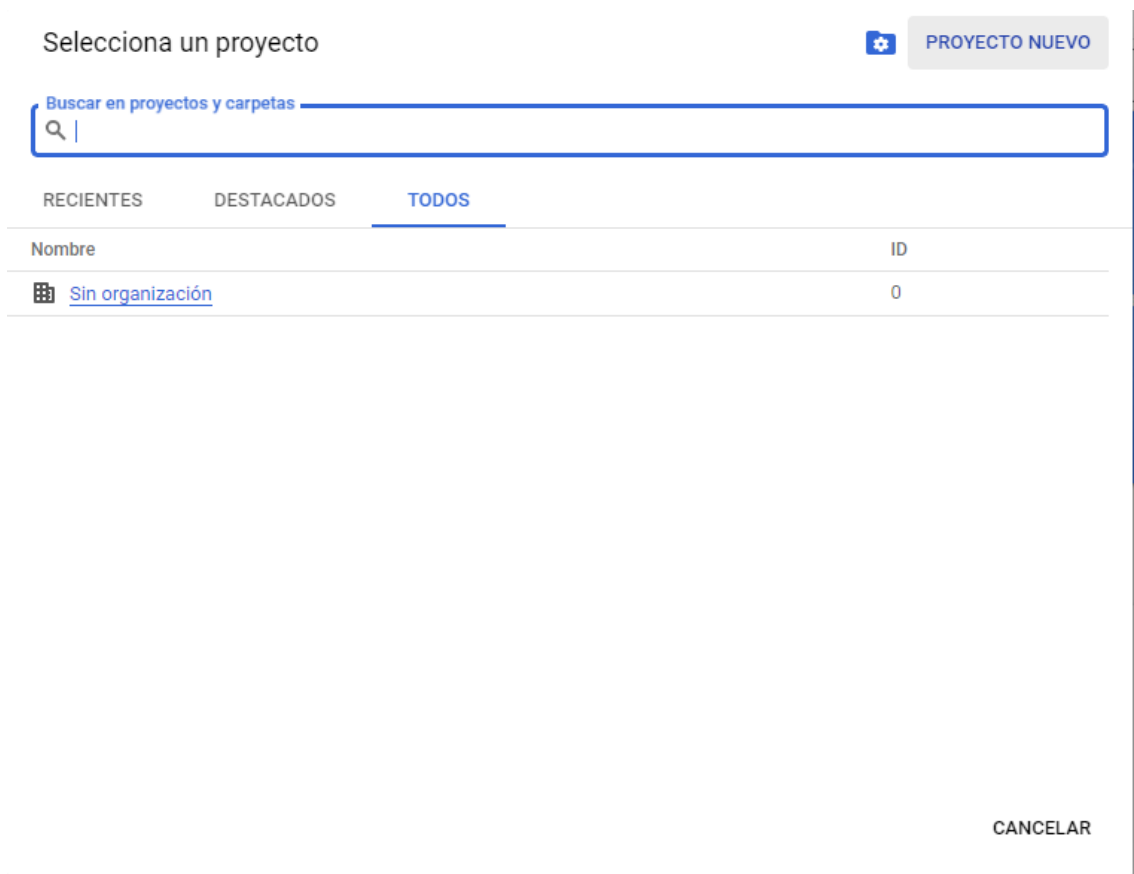
<https://drive.google.com/drive/folders/1WZbU6FwuU5SIMST5uDKXMY-VFXWPNS9n>

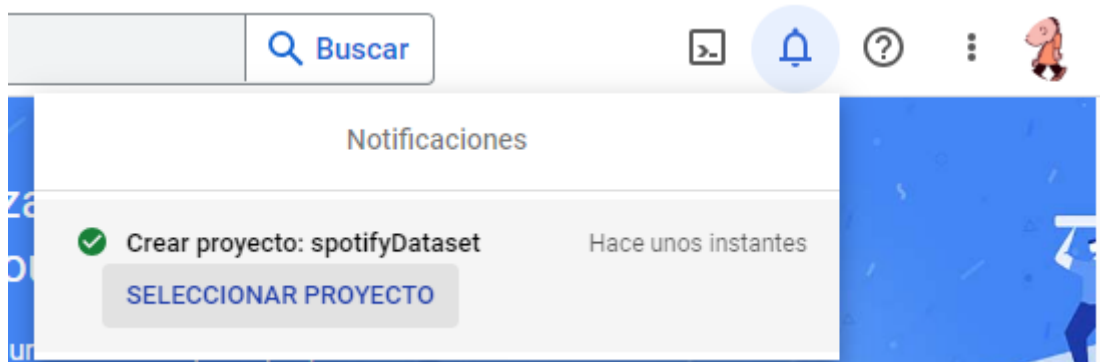
Google Drive

Para poder subir nuestro archivo merged.csv a traves de codigo, tenemos que usar la plataforma de google cloud para poder hacerlo, entonces creamos un usuario y nos logueamos, aceptando terminos y condiciones.

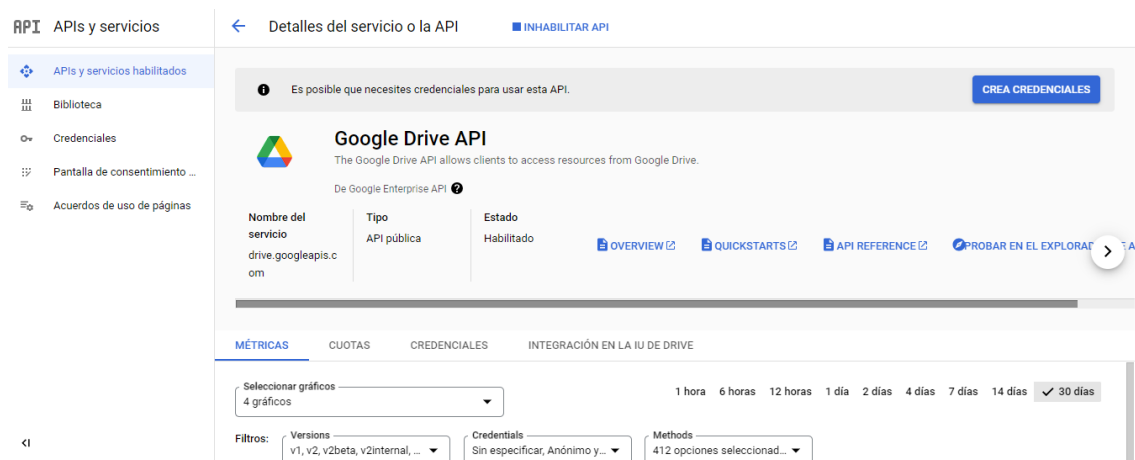


Debemos crear un nuevo proyecto, el cual llamamos **spotifyDataset**

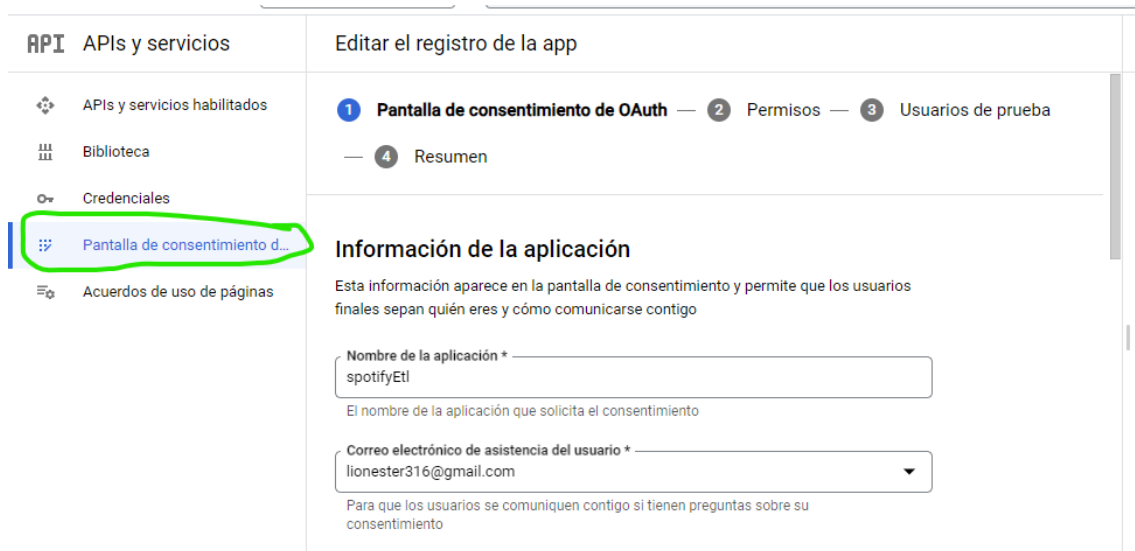




Buscamos dentro de GCP **Google Drive API**



En este caso usaremos OAuth y registramos nuestra app en la Pantalla de Consentimiento.



Copiamos el link de la barra de búsqueda y lo ponemos en esta seccion:

Dominio de la app

Para protegerlos a ti y a tus usuarios, Google solo permite que las apps que usan OAuth puedan emplear los dominios autorizados. Se mostrará la siguiente información a los usuarios en la pantalla de consentimiento.

Página principal de la aplicación

<https://console.cloud.google.com/apis/credentials/consent/edit;newAppInternalUser>

Proporciona a los usuarios un vínculo a tu página principal

Vínculo a la Política de Privacidad de la aplicación

<https://console.cloud.google.com/apis/credentials/consent/edit;newAppInternalUser>

Proporciona a los usuarios un vínculo a tu página pública de Política de Privacidad

Vínculo a las Condiciones del Servicio de la aplicación

<s/credentials/consent/edit;newAppInternalUser=false?project=spotifydataset-402017>

Proporciona a los usuarios un vínculo a tu página pública de Condiciones del Servicio

Colocamos nuestro correo y le damos Guardar y continuar

RPI	APIs y servicios	Editar el registro de la app
	APIs y servicios habilitados	Proporciona a los usuarios un vínculo a tu página pública de Condiciones del Servicio
	Biblioteca	
	Credenciales	
	Pantalla de consentimiento d...	
	Acuerdos de uso de páginas	
		Dominios autorizados ?
		Cuando un dominio se usa en la pantalla de consentimiento o en la configuración del cliente de OAuth, debe contar con un registro previo aquí. Si debes verificar la app, ve Google Search Console para comprobar si tus dominios están autorizados. Más información
		Dominio autorizado 1 * <input type="text" value="google.com"/>
		+ AGREGAR UN DOMINIO
		Información de contacto del desarrollador
		Direcciones de correo electrónico * <input type="text" value="lionester316@gmail.com"/>
		Google enviará notificaciones sobre cualquier cambio en tu proyecto a estas direcciones de correo electrónico.
		GUARDAR Y CONTINUAR CANCELAR

- ✓ Pantalla de consentimiento de OAuth — 2 **Permisos** — 3 Usuarios de prueba — 4 Resumen

En esta seccion de **Permisos** no configuramos nada entonces le damos siguiente

En la seccion de usuarios de prueba ponemos nuestro correo:

✕

Agregar usuarios

Mientras el estado de publicación sea "Prueba", solo los usuarios de prueba podrán acceder a la app. El límite de usuarios permitidos antes de que se verifique la app es de 100, y se calcula según el ciclo de vida completo de la app.

LEARN MORE [↗](#)

lionester316@gmail.com ✕

?

1 / 100

AGREGAR

Filtro

Ingresar el nombre o el valor de la propiedad

?

Información del usuario	
lionester316@gmail.com	<div></div>

GUARDAR Y CONTINUAR

CANCELAR

Y asi quedaron las Configuraciones:

Editar el registro de la app

Nombre de la app

spotifyEtl

Correo electrónico de asistencia

lionester316@gmail.com

Logotipo de la app

No se proporcionó

Vínculo a la página principal de la aplicación

https://console.cloud.google.com/apis/credentials/consent/edit;newAppInternalUser=false?project=spotifydataset-402017

Vínculo a la Política de Privacidad de la aplicación

https://console.cloud.google.com/apis/credentials/consent/edit;newAppInternalUser=false?project=spotifydataset-402017

Vínculo a las Condiciones del Servicio de la aplicación

https://console.cloud.google.com/apis/credentials/consent/edit;newAppInternalUser=false?project=spotifydataset-402017

Ahora procedemos a la Creacion de las Credenciales, Nos conectaemos atraves de el ID de cliente de OAuth.

RPI

APIs y servicios

APIs y servicios habilitados

Biblioteca

Credenciales

Pantalla de consentimiento ...

Acuerdos de uso de páginas

Credenciales

+ CREAR CREDENCIALES

🗑 BORRAR

↩ RESTABLECER CREDENCIALES BORRADAS

Crea credenciales para acceso

Clave de API

ID de cliente de OAuth

Cuenta de servicio

Claves de API

☐

Nombre

No hay claves de API para mostrar

ID de clientes OAuth

☐

Nombre

No hay clientes de OAuth para mostrar

Cuentas de servicio

☐

Correo electrónico

No hay cuentas de servicio para mostrar

Clave de API

Identifica tu proyecto con una clave de API simple para verificar la cuota y el acceso

ID de cliente de OAuth

Solicita el consentimiento del usuario para que tu app pueda acceder a sus datos

Cuenta de servicio

Habilita la autenticación de servidor a servidor en el nivel de la app mediante cuentas robot

Ayúdame a elegir

Responde algunas preguntas para decidir qué tipo de credencial usar

iones

Acciones

ID de cliente

Acciones

Administrar cuentas de servicio

Acciones

Empezamos con las Configuraciones:

←

Crear ID de cliente de OAuth

de Google. Si la app se ejecuta en varias plataformas, cada una necesitará su propio ID de cliente. Consulta [Configura OAuth 2.0](#) para obtener más información. [Obtén más información](#) sobre los tipos de clientes de OAuth.

Tipo de aplicación *

Aplicación web

Nombre *

spotifyDataset

El nombre de tu cliente de OAuth 2.0. Este nombre solo se usa para identificar al cliente en la consola y no se mostrará a los usuarios finales.

i

Los dominios de los URI que agregues a continuación se incorporarán automáticamente a tu [pantalla de consentimiento de OAuth](#) como [dominios autorizados](#).

Orígenes autorizados de JavaScript

?

Para usar con solicitudes de un navegador

URI 1 *

http://localhost:1080

CORRECCION: En la URL es <http://localhost:8080>

Le damos crear y descargamos el JSON.

Se creó el cliente de OAuth

Puedes acceder al ID de cliente y el secreto desde "Credenciales" en API y servicios

i

El acceso OAuth está restringido a los [usuarios de prueba](#) que aparecen en la [pantalla de consentimiento de OAuth](#)

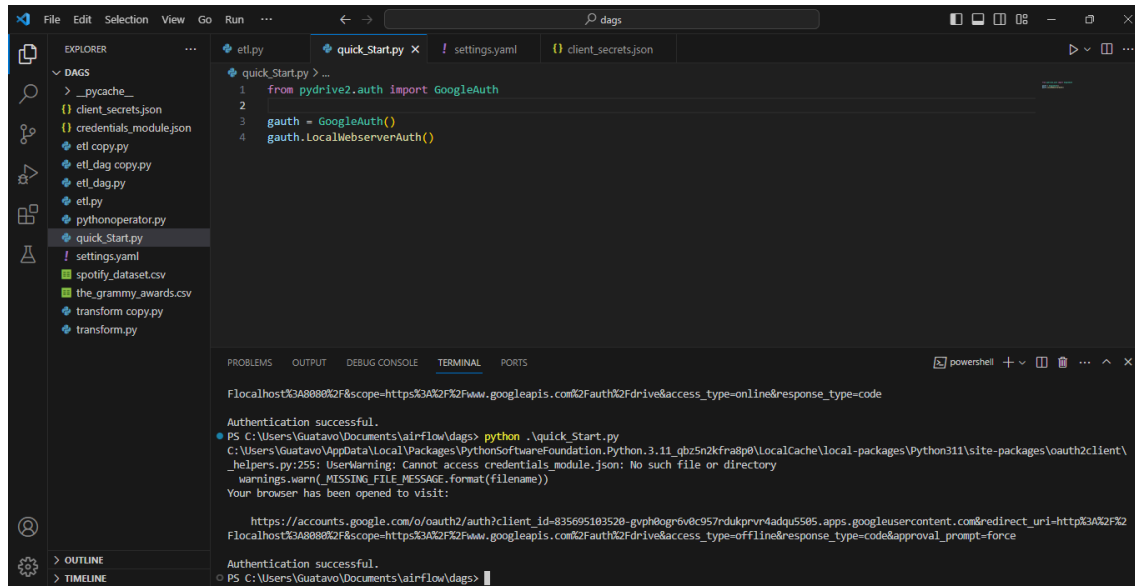
ID de cliente	835695103520-gvph0ogr6v0c957rdukprvr4adqu5505.apps.googleusercontent.com
Secreto del cliente	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
Fecha de creación	14 de octubre de 2023, 12:11:01 GMT-5
Estado	<div>✓</div> Habilitada

↓

 DESCARGAR JSON

ACEPTAR

Creamos un archivo **quick_Start.py** el cual nos permitira autenticarnos.



```
1 from pydrive2.auth import GoogleAuth
2
3 gauth = GoogleAuth()
4 gauth.LocalWebserverAuth()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

FlocalhostX3A8888X2F&scope=httpsX3A3AX2F2Fwww.googleapis.comX2FauthX2Fdrive&access_type=online&response_type=code

Authentication successful.

PS C:\Users\Guatavo\Documents\airflow\dags> python .\quick_Start.py

C:\Users\Guatavo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfr8p0\LocalCache\local-packages\Python311\site-packages\oauth2client\helpers.py:255: UserWarning: Cannot access credentials module.json: No such file or directory

warnings.warn(_MISSING_FILE_MESSAGE.format(filename))

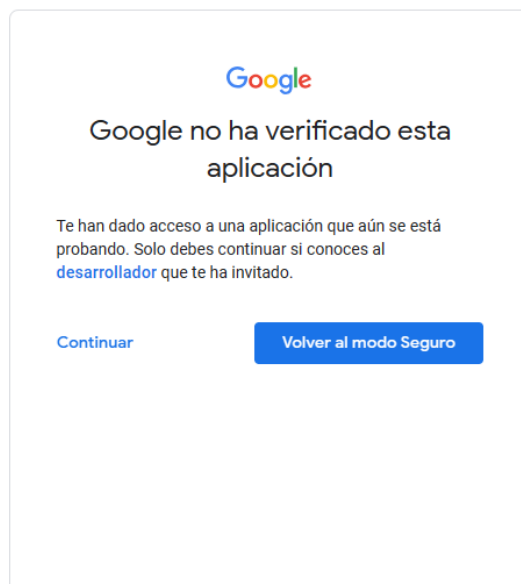
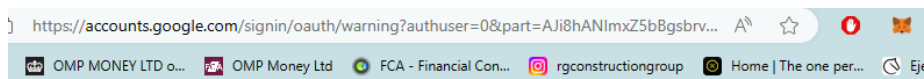
Your browser has been opened to visit:

https://accounts.google.com/o/oauth2/auth?client_id=835695183528-gvph8ogr6vc957rdukprvr4adqu5585.apps.googleusercontent.com&redirect_uri=httpX3A3AX2F2FlocalhostX3A8888X2F&scope=httpsX3A3AX2F2Fwww.googleapis.comX2FauthX2Fdrive&access_type=offline&response_type=code&approval_prompt=force

Authentication successful.

PS C:\Users\Guatavo\Documents\airflow\dags> |

Corremos el archivo, nos lleva a la web donde le damos continuar



nos autenticamos iniciando sesión.

spotifyEtl quiere acceder a tu cuenta de Google



lionester316@gmail.com

Cuando permitas este acceso, **spotifyEtl** podrá:



Ver, modificar, crear y eliminar archivos de Google Drive. [Más información](#)

Confirma que confías en spotifyEtl

Puede que estés compartiendo información sensible con este sitio o esta aplicación. Puedes ver o retirar el acceso en cualquier momento en tu [cuenta de Google](#).

Descubre cómo te ayuda Google a [compartir datos de forma segura](#).

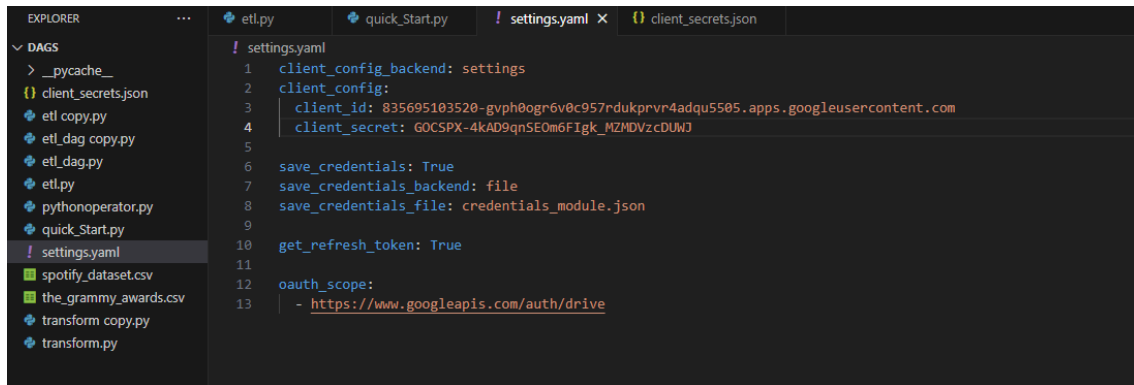
Consulta la [Política de Privacidad](#) y los [Términos del Servicio](#) de spotifyEtl.

Cancelar

Continuar

The authentication flow has completed.

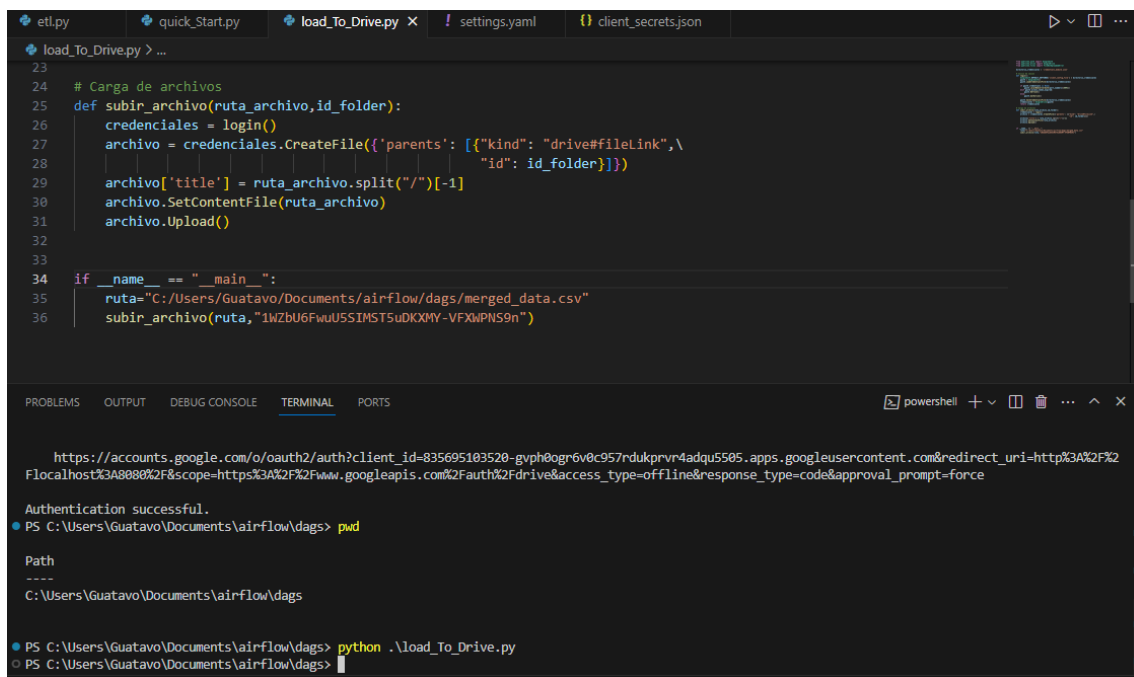
Luego hacemos unas configuraciones para crear persistencia, es decir para que no nos pida autenticarnos cada vez que queramos subir un archivo. Debemos recordar que El `client_id` y el `client_secret` los sacamos de la app que creamos en GCP



The screenshot shows the VS Code Explorer on the left with a file tree containing various Python scripts and configuration files. The main editor displays the `settings.yaml` file with the following content:

```
1 client_config_backend: settings
2 client_config:
3   client_id: 835695103520-gvph0ogr6v8c957rdukprvr4adqu5505.apps.googleusercontent.com
4   client_secret: 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
5
6 save_credentials: True
7 save_credentials_backend: file
8 save_credentials_file: credentials_module.json
9
10 get_refresh_token: True
11
12 oauth_scope:
13   - https://www.googleapis.com/auth/drive
```

Corremos el archivo `load_to_Drive.py` que tiene 2 funciones, `login()` que nos permite loguearnos y crear la conexión con Google Drive y `subir_archivo()` que sube el archivo recibiendo los parametros (`ruta_del_archivo_a_subir`, `id_de_la_carpeta`)



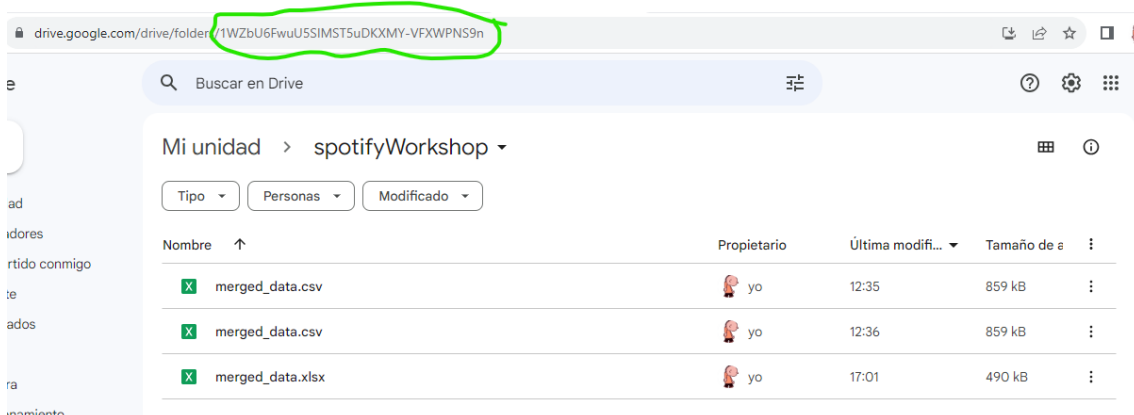
The screenshot shows the VS Code editor with the `load_To_Drive.py` file open. The code defines a `subir_archivo` function that uses `login` and Google Drive API to upload a file. Below the code, the terminal window shows the execution of the script, including successful authentication and the current directory path.

```
23
24 # Carga de archivos
25 def subir_archivo(ruta_archivo,id_folder):
26     credenciales = login()
27     archivo = credenciales.CreateFile({'parents': [{"kind": "drive#fileLink",\
28                                               "id": id_folder}]})
29     archivo['title'] = ruta_archivo.split("/")[-1]
30     archivo.SetContentFile(ruta_archivo)
31     archivo.Upload()
32
33
34 if __name__ == "__main__":
35     ruta="C:/Users/Guatavo/Documents/airflow/dags/merged_data.csv"
36     subir_archivo(ruta,"1WZbU6FwuU5SIMST5uDKXMY-VFXWPNS9n")
```

Terminal output:

```
https://accounts.google.com/o/oauth2/auth?client_id=835695103520-gvph0ogr6v8c957rdukprvr4adqu5505.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2F&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive&access_type=offline&response_type=code&approval_prompt=force
Authentication successful.
PS C:\Users\Guatavo\Documents\airflow\dags> pwd
Path
----
C:\Users\Guatavo\Documents\airflow\dags
PS C:\Users\Guatavo\Documents\airflow\dags> python .\load_To_Drive.py
PS C:\Users\Guatavo\Documents\airflow\dags>
```

Corremos el archivo, y efectivamente podemos ver de que se subio el archivo.



Lo que esta en verde es el `id_de_la_carpeta`.

<https://drive.google.com/drive/folders/1WZbU6FwuU5SIMST5uDKXMY-VFXWPNS9n>