WORKSHOP - 002

Paola Andrea Chaux Campo - 2220225

ETL (Extracción, Transformación y carga)

Docente:
Javier Alejandro Vergara

Ingeniería de Datos e Inteligencia Artificial
Facultad de Ingeniería
Universidad Autónoma De Occidente
2024

Documentation Workshop 002

In the Airflow Data Engineer code challenge. I show you my knowledge about data management and visualizations with the final objective of shows all the ETL process using the two different data sources (csv and Database) and chart visualizations in Power BI. In this workshop I used the spotify dataset (csv) to be readed in python and airflow, create some transformation and load into a database, on the other hand, I used the grammys dataset to be loaded into a database, then using Airflow I readed the data from the database , perform transformations, merge with the spotify dataset and load into the database.

Paso 1:

First of all, we started with the EDA of each file to determine the proper transformations that will be made. It was taken into account that the objective of this analysis was to identify which musical genres are most successful at the Grammys, the impact or relationship of the awards and the popularity of Spotify.:

SPOTIFY (CSV)
For the data in Spotify, the data was first read as csv.

```
spotify = pd.read_csv('P:/ETL/workshop_002/Data/spotify_dataset.csv', sep=",")
spotify.head()
```

Python

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mod
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.461	1	-6.746	
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.166	1	-17.235	
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid	To Begin	To Begin	57	210826	False	0.438	0.359	0	-9.734	

Delete the first column that was not necessary and is one of the transformations performed in airflow, I review the attributes and the shape.

```
We can see that the atribute "Unnamed" isn't important, this wil be eliminated

spotify = spotify.drop('Unnamed: 0', axis=1)
```

Python

```
spotify.columns
```

Python

```
Index(['track_id', 'artists', 'album_name', 'track_name', 'popularity', 'duration_ms', 'explicit', 'danceability', 'energy', 'key', 'loudness', 'mode',
```

```
spotify.shape
```

Python

```
(114000, 20)
```

Our dataset has a dimension of (114000, 21), i.e. 114,000 songs, and consists of 21 columns or atributes.

```
spotify.nunique()
```

```
track_id            89741
artists             31437
album_name          46589
track_name          73608
popularity            101
duration_ms         50697
explicit                2
danceability         1174
energy               2083
key                    12
loudness            19480
mode                    2
speechiness          1489
acousticness         5061
instrumentalness     5346
liveness             1722
valence              1790
tempo               45653
time_signature          5
track_genre           114
dtype: int64
```

I review the unique values to profile the dataset.

```
spotify.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        114000 non-null  int64  
 1   track_id          114000 non-null  object 
 2   artists           113999 non-null  object 
 3   album_name        113999 non-null  object 
 4   track_name        113999 non-null  object 
 5   popularity        114000 non-null  int64  
 6   duration_ms       114000 non-null  int64  
 7   explicit          114000 non-null  bool   
 8   danceability      114000 non-null  float64
 9   energy            114000 non-null  float64
 10  key               114000 non-null  int64  
 11  loudness          114000 non-null  float64
 12  mode              114000 non-null  int64  
 13  speechiness       114000 non-null  float64
 14  acousticness      114000 non-null  float64
 15  instrumentalness  114000 non-null  float64
 16  liveness          114000 non-null  float64
 17  valence           114000 non-null  float64
 18  tempo             114000 non-null  float64
 19  time_signature    114000 non-null  int64  
 20  track_genre       114000 non-null  object 
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 17.5+ MB
```

Here, we must take into account that the attribute "Move" is boleean but it's represented as int64, this will be taken into account and changed.

I check the null values since they cannot have for my objective

```
spotify.isnull().sum()
```

```
track_id            0
artists             1
album_name          1
track_name          1
popularity          0
duration_ms         0
explicit            0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
track_genre         0
dtype: int64
```

Here, we can see that we are missing an artist, album name and track name, let's review

```
review = spotify[spotify.isnull().any(axis=1)]
review.head()
```

Python

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...	loudness	mode	speec
65900	65900	1kR4glb7nGxHPI3D2ifs59	NaN	NaN	NaN	0	0	False	0.501	0.583	...	-9.46	0	

1 rows × 21 columns

```
spotify[spotify['duration_ms'] == 0]
```

Python

	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	ac
65900	1kR4glb7nGxHPI3D2ifs59	NaN	NaN	NaN	0	0	False	0.501	0.583	7	-9.46	False	0.06	

When I check it, we can see, the valors that are missings, all are the same row, in additions, this row contain a duration of 0 ms. This will be eliminated

DUPLICATED

```
id_duplicated = spotify.duplicated(subset=['track_id']).sum()
print("Numero de id de las canciones duplicadas:", id_duplicated)

name_duplicated = spotify.duplicated(subset=['track_name']).sum()
print("Numero de nombres de las canciones duplicadas:", name_duplicated)
```

```
Numero de id de las canciones duplicadas: 24259
Numero de nombres de las canciones duplicadas: 40391
```

I check the duplicates and there are quite a few, values that should not be there.
We have identified 24,259 songs with the same track_id, plus a total of 40,391 songs with the same name in our dataset. I check this.

```python
spotify_copy = spotify.copy()
spotify_copy['track_name'] = spotify_copy['track_name'].str.lower()
grouped_df = (spotify_copy.groupby(['track_name', 'artists'])
                          .size()
                          .reset_index(name='count')
                          .sort_values(by='count', ascending=False))
print(grouped_df)
```

	track_name	artists	count
56404	run rudolph run	Chuck Berry	151
38577	little saint nick - 1991 remix	The Beach Boys	76
37077	last last	Burna Boy	75
24351	frosty the snowman	Ella Fitzgerald	69
12284	christmas time	Bryan Adams	66
14714	cómo se siente - remix	Jhayco;Bad Bunny	64
60186	sleigh ride	Ella Fitzgerald	60
56332	rumbatón	Daddy Yankee	60
75684	x última vez	Daddy Yankee;Bad Bunny	58
22662	feliz cumpleaños ferxxo	Feid	54
37941	ley seca	Jhayco;Anuel AA	54
29464	hot in it	Tiësto;Charli XCX	54
48540	on repeat	Robin Schulz;David Guetta	52
53839	qué más pues?	J Balvin;Maria Becerra	51
12226	christmas all over again	Tom Petty and the Heartbreakers	47
27290	happier	Marshmello;Bastille	46
23105	first class	Jack Harlow	45
55898	rockin' around the christmas tree	Brenda Lee	45
50007	pantysito	Alejo;Feid;ROBI	44
57951	secreto	Anuel AA;KAROL G	44
74056	what christmas means to me - single version / ...	Stevie Wonder	44

After grouping songs by track name and artists, we found 81,206 unique combinations.

```python
spotify_c = spotify.copy()
spotify_c['track_name'] = spotify_c['track_name'].str.lower()

duplicates = spotify_c.duplicated(subset=['track_id', 'track_name'], keep=False)
spotify_duplicates = spotify_c[duplicates]

grouped_duplicates = (spotify_duplicates.groupby(['track_id', 'track_name', 'artists'])
                                        .size()
                                        .reset_index(name='count')
                                        .sort_values(by='count', ascending=False))

filtered_duplicates = grouped_duplicates[grouped_duplicates['count'] >= 2]

print(f"Número total de registros mostrados: {filtered_duplicates.shape[0]}")
for index, row in filtered_duplicates.iterrows():
    print(f"Track ID: {row['track_id']}, Track Name: {row['track_name']}, Artist: {row['artists']}, Count: {row['count']}")
```

```
Número total de registros mostrados: 16641
Track ID: 6S3JlDAGk3uu3NtZbPnuhS, Track Name: baby blue - remastered 2010, Artist: Badfinger, Count: 9
Track ID: 2kkvB3RNRzwjFdGhaUA0tz, Track Name: layla, Artist: Derek & The Dominos, Count: 8
Track ID: 2Ey6v4Sekh3Z0RUSISRosD, Track Name: layla, Artist: Derek & The Dominos, Count: 8
Track ID: 4WJTKbNJQ41zXnb84jSWaj, Track Name: roots bloody roots, Artist: Sepultura, Count: 7
Track ID: 2vU6bm5hVF2idVknGzqyPL, Track Name: christmas (baby please come home), Artist: The Offspring, Count: 7
Track ID: 5ftfVzSLIi5ZxYdNbRtf41, Track Name: never gonna give you up, Artist: The Black Keys, Count: 7
Track ID: 5BI1XqMJK91dsEq0Bfe0Ov, Track Name: show me the way, Artist: Peter Frampton, Count: 7
Track ID: 4aqS25F3ywJ9TGnNkOqilC, Track Name: abracadabra - remastered 2017, Artist: Steve Miller Band, Count: 7
Track ID: 08kTa3SL9sV6Iy8KLKtGql, Track Name: kasoor, Artist: Prateek Kuhad, Count: 7
Track ID: 5ZsAhuQ24mWHiduaxJqnhW, Track Name: udd gaye, Artist: Ritviz, Count: 7
Track ID: 36NwMJRaCy7x77xYGJiG2M, Track Name: midnight rider, Artist: Allman Brothers Band, Count: 7
Track ID: 4XYieGKSlJlHpzB3bl6WMP, Track Name: abracadabra - remastered 2017, Artist: Steve Miller Band, Count: 7
```

Here what I do is identify the songs that are duplicated by id and name and I group them by id, name and artist to check how many songs are repeated, identical and we should eliminate, this gives me a total of 16,641 records which we will proceed to eliminate and leave only the most popular since I will focus on analyzing some evaluation between popularity and winning awards also having a differentiation by gender

```
spotify['track_name'] = spotify['track_name'].str.lower()

indices_max_popularity = spotify.groupby(['track_id', 'track_name', 'artists'])['popularity'].idxmax()
spotify = spotify.loc[indices_max_popularity]
spotify.reset_index(drop=True, inplace=True)
print(spotify)
num_rows = spotify.shape[0]
print("Número de filas en el DataFrame:", num_rows)
```

```
                track_id                         artists                                album_name
0      0000vdREvCVMxbQTkS888c                            Rill                                 Lolly
1      000CC8EParg64OmTxVnZ0p                       Glee Cast                  Glee Love Songs  it's all comi
2      000Iz0K615UepwSJ5z2RE5              Paul Kalkbrenner;Pig&Dan                                 X
3      000RDCYioLteXcutOjeweY                     Jordan Sandhu                           Teeje Week
4      000qpdoc97IMTBvF8gwcpy                    Paul Kalkbrenner                                Zeit
5      0017XiMkqbTfF2AUOzlhj6                       Chad Daniels                  Busy Being Awesome
6      001APMDOl3qtx1526T11n1                    Pink Sweat$;Kirby                             New RnB
7      001YQlnDSduXd5LgBd66gT                       Soda Stereo              Soda Stereo (Remastered)          el t
8      001pyq8FLNSL1C8orNLI0b                Old Crow Medicine Show                           O.C.M.S.
9      002qpSULhHAw6DGqFxbaO1                   Tokyo Ghetto Pussy                          Disco 2001
10     002uYDBLOvJz21C4FuArDS                       Sigma;Birdy                      Find Me (Remixes)
11     003lo4y8gOylAqDs2scLx2                            Dither            Dominator - We Will Prevail
12     003vvx7Niy0yvhvHt4a68B                       The Killers                             Hot Fuss
13     004G9E3EZhxxn5aE9yEQqx                     Pappo's Blues                 Pappo's Blues, Vol. 3
14     004h8smbIoAkUNDJvVKwkG                        Ouse;Powfu                          Loners Diary
15     004iWPkSRbvOEvAPLWHl9M  Ernest Tubb;The Wilburn Brothers                      Definitive Hits
16     0051nJ5xbRu8kuqPTYa9l7                          Pigface                 The Best Of Pigface
17     005NLlv3rClEVqeoSyYTHU                            Leola                Things change but not all
```

The songs that were duplicates were eliminated and a total of 89,740 rows remained that are clean.

```
spotify_c1 = spotify.copy()
spotify_c1['track_name'] = spotify_c1['track_name'].str.lower()

duplicates1 = spotify_c1.duplicated(subset=['track_id', 'track_name'], keep=False)
spotify_duplicates1 = spotify_c1[duplicates1]

grouped_duplicates1 = (spotify_duplicates1.groupby(['track_id', 'track_name', 'artists'])
                       .size()
                       .reset_index(name='count')
                       .sort_values(by='count', ascending=False))

filtered_duplicates1 = grouped_duplicates1[grouped_duplicates1['count'] >= 2]

print(f"Número total de registros mostrados: {filtered_duplicates1.shape[0]}")
for index, row in filtered_duplicates1.iterrows():
    print(f"Track ID: {row['track_id']}, Track Name: {row['track_name']}, Artist: {row['artists']}, Count: {row['count']}")
```

```
Número total de registros mostrados: 0
```

We don't have any rows of duplicated for id, name and artist. Now, we check that no have duplicated song for name and artist.

```
spotify_prueba= spotify.copy()
spotify_prueba['track_name'] = spotify_prueba['track_name'].str.lower()

duplicatesp = spotify_prueba.duplicated(subset=[ 'track_name'], keep=False)
spotify_duplicates_prueba = spotify_prueba[duplicatesp]

grouped_duplicates_prueba = (spotify_duplicates_prueba.groupby([ 'track_name', 'artists'])
                       .size()
                       .reset_index(name='count')
                       .sort_values(by='count', ascending=False))

filtered_duplicatesp = grouped_duplicates_prueba[grouped_duplicates_prueba['count'] >= 2]

print(f"Número total de registros mostrados: {filtered_duplicatesp.shape[0]}")
for index, row in filtered_duplicatesp.iterrows():
    print(f"Track Name: {row['track_name']}, Artist: {row['artists']}, Count: {row['count']}")
```

```
Número total de registros mostrados: 4747
Track Name: rockin' around the christmas tree, Artist: Brenda Lee, Count: 45
Track Name: little saint nick - 1991 remix, Artist: The Beach Boys, Count: 41
Track Name: run rudolph run, Artist: Chuck Berry, Count: 40
Track Name: frosty the snowman, Artist: Ella Fitzgerald, Count: 34
Track Name: let it snow! let it snow! let it snow!, Artist: Dean Martin, Count: 32
Track Name: mistletoe, Artist: Justin Bieber, Count: 31
Track Name: sleigh ride, Artist: Ella Fitzgerald, Count: 30
Track Name: i saw mommy kissing santa claus, Artist: The Jackson 5, Count: 27
Track Name: the christmas song (merry christmas to you), Artist: Nat King Cole, Count: 26
Track Name: last last, Artist: Burna Boy, Count: 26
Track Name: santa claus is coming to town, Artist: The Jackson 5, Count: 26
Track Name: cozy little christmas, Artist: Katy Perry, Count: 25
Track Name: 21 reasons, Artist: Nathan Dawe;Ella Henderson, Count: 25
Track Name: make it to christmas, Artist: Alessia Cara, Count: 24
Track Name: first class, Artist: Jack Harlow, Count: 23
```

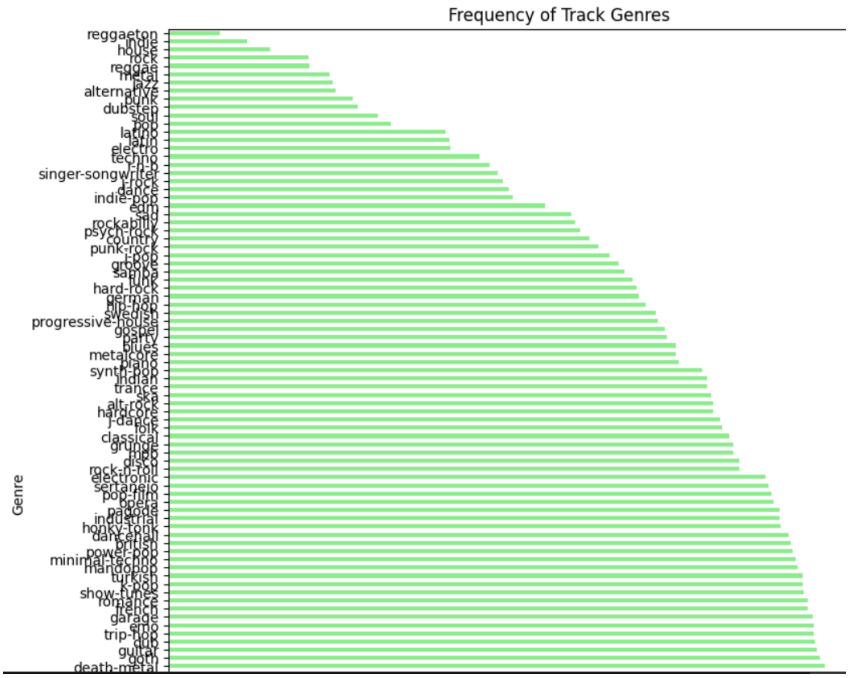
```
indices_max_popularity2 = spotify.groupby(['track_name', 'artists'])['popularity'].idxmax()

spotify = spotify.loc[indices_max_popularity2]
spotify.reset_index(drop=True, inplace=True)

print(spotify)
num_rows = spotify.shape[0]
print("Numero de filas:", num_rows)
```

	track_id	artists	album_name	
0	0fROT4kK5oTm8xO8PX6EJF	Rilès	!I'll Be Back!	
1	1hH0t381PIXmUVWyG1Vj3p	Brian Hyland	The Bashful Blond	
2	1B45DvGMoFWdbAEUH2qliG	Little Apple Band	The Favorite Songs Of Sesame Street	
3	0jmz4aHEIBCRgrcV2xEkwB	Traditional;Sistine Chapel Choir;Massimo Palom...	Classical Christmas	
4	5Zx0Rrkn5RFBMD2PRxX3mI	Dillinger Four	C I V I L W A R	"co
5	3ozivYJGJGq6TSzdy8m64X	Capcom Sound Team	デビル メイ クライ 3 オリジナル・サウンドトラッ	
6	3KKk48f33mlB56F5L5nbJk	Nikolay Kopylov	Popular Opera Arias	
7	5OiONTndVC5YOMXg6VC5xs	Nikolay Kopylov	Popular Opera Arias	
8	4gHZlq1u5m89HP96BT3QHk	Dave Brubeck	A Dave Brubeck Christmas	
9	6ZZU6LPjm5ueOsWHb6X5cA	Hans Zimmer;Lisa Gerrard;Klaus Badelt;Tamara T...	Hans Zimmer: Epic Scores	"gladiator" - m
10	4yfFraZrNnh2zJTok5fzq7	Felix Mendelssohn;Christopher Herrick;Simon Pr...	Klassische Weihnachtsmusik	
11	5ezuHIXXlsPQY4rbsSKT1W	Laura Osnes	Rodgers + Hammerstein's Cinderella (Original B...	
12	0VXg0YYguSikxQCZSELGdX	Laura Osnes;Victoria Clark	Rodgers + Hammerstein's Cinderella (Original B...	
13	14Qcrx6Dfjvcj0H8oV8oUW	London Symphony Orchestra	The London Symphony Orchestra: The Top 100 of ...	"in the hall o
14	5NvjrzzqYEvOg7U189N89W	Michael Cerveris;Beth Malone	Fun Home (A New Broadway Musical)	
15	6KNGqIRhE44Cf56dvcgi1l	Inspector	Ska a La Carta	
16	280zqg7gblMTRU1Hc5VxpY	Santino Fontana;Laura Osnes	Rodgers + Hammerstein's Cinderella (Original B...	
17	7lrpT3dtXvYhTkLceXlDUx	Michael Cerveris;Sydney Lucas;Beth Malone;Judy...	Fun Home (A New Broadway Musical)	

```
df = spotify.groupby(['track_name', 'artists']).size().reset_index(name='count')
df.sort_values(by = 'count', ascending= False)
print(df)
num_rows = spotify.shape[0]
print("Numero de filas:", num_rows)
```

	track_name	artists	count
0	!i'll be back!	Rilès	1
1	"a" you're adorable	Brian Hyland	1
2	"c" is for cookie	Little Apple Band	1
3	"christe, redemptor omnium"	Traditional;Sistine Chapel Choir;Massimo Palom...	1
4	"contemplate this on the tree of woe."	Dillinger Four	1
5	"devils never cry"(スタッフロール)	Capcom Sound Team	1
6	"don carlos" roderigo's death aria	Nikolay Kopylov	1
7	"eugene onegin" ariozo of onegin	Nikolay Kopylov	1
8	"farewell" jingle bells	Dave Brubeck	1
9	"gladiator" - music from the motion picture: n...	Hans Zimmer;Lisa Gerrard;Klaus Badelt;Tamara T...	1
10	"hark! the herald angels sing"	Felix Mendelssohn;Christopher Herrick;Simon Pr...	1
11	"he was tall"	Laura Osnes	1
12	"in my own little corner" - reprise	Laura Osnes;Victoria Clark	1
13	"in the hall of the mountain king" from peer g...	London Symphony Orchestra	1
14	"it was great to have you home..."	Michael Cerveris;Beth Malone	1
15	"lamba do ska" (llorando se fue)	Inspector	1
16	"loneliness of evening"	Santino Fontana;Laura Osnes	1
17	"read a book..."	Michael Cerveris;Sydney Lucas;Beth Malone;Judy...	1
18	"shortly after we were married..."	Judy Kuhn;Emily Skeggs	1
19	"something in the rain" (something in the rain...	Rachael Yamagata	1
20	"was he slow?" - music from the motion picture...	Kid Koala	1
21	"why is a raven like a writing desk?"	arai tasuku	1
22	# 1	Slank	1

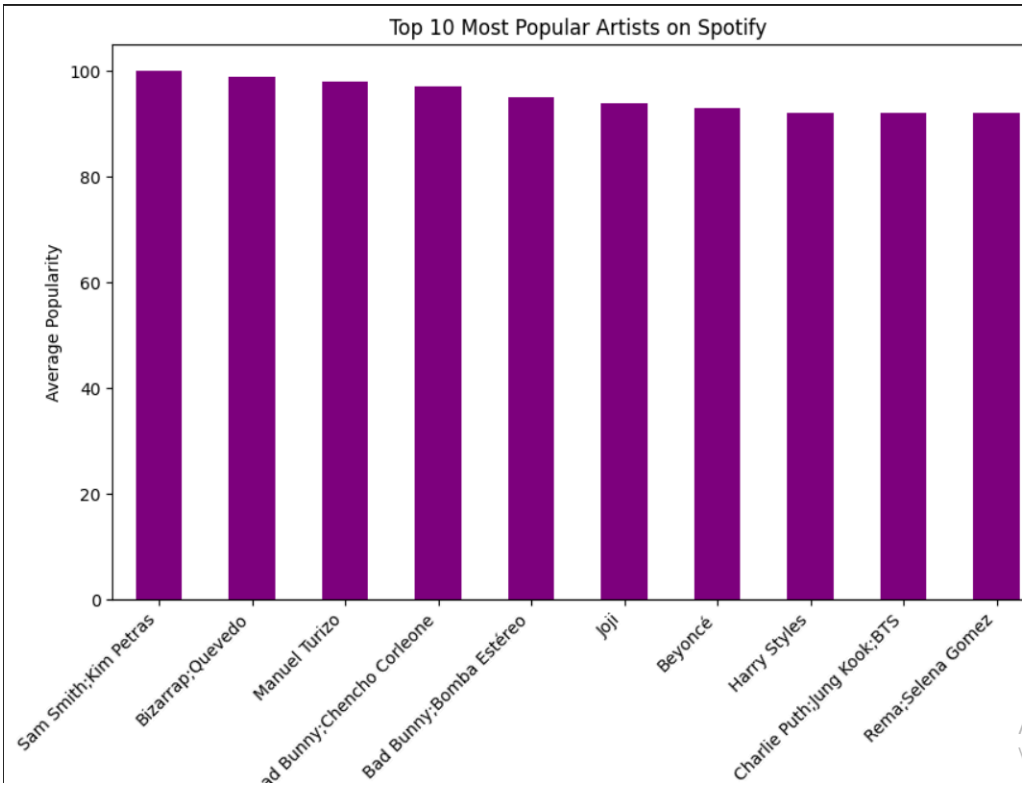
This would be my final dataset with a total of 81,206 rows with the clean data that I will use.



Here we can see the frequencies by genre from the least listened to to the most listened to.

Genre	Frequency
study	996
black-metal	990
comedy	987
heavy-metal	981
bluegrass	976
forro	965
malay	964
grindcore	963
idm	958
iranian	955
cantopop	955
chicago-house	954
new-age	954
breakbeat	954
afrobeat	952
club	946
sleep	945
disney	944
happy	943
anime	942
acoustic	937
kids	937
...	
house	136
indie	105
reggaeton	68

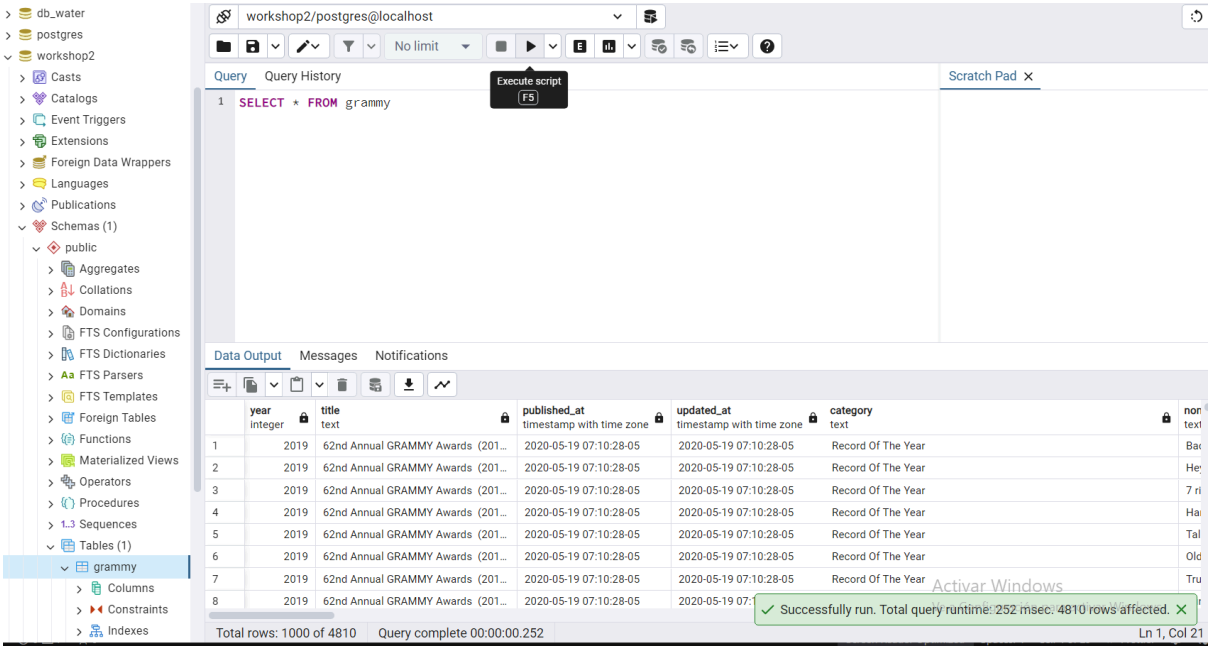
Here, we can see that the most frecuent genres are: study (996), balck-metal (990), comedy (987), heavy-metal (981) and the least frecuent to are: house (136), indie (105), reggaeton (68).



Here we have the top 10 of the most popular artists on Spotify.

GRAMMY (Database)

With this dataset, the first thing that was done was to upload it to the database for the workshop since we had it in csv, here they are already in the database.



The due EDA was carried out:

the call was made to the data in the database

```python
def extract_data(filename, db_name, table_name):
    connection = None
    try:
        with open(filename, 'r') as file:
            config = json.load(file)

        connection = psycopg2.connect(
            host='localhost',
            user=config["user"],
            password=config["password"],
            dbname=db_name
        )
        query = f"SELECT * FROM {table_name}"
        df = pd.read_sql(query, connection)

        return df

    except (Exception, psycopg2.DatabaseError) as error:
        print(f"Error al consultar datos: {error}")
        return None

    finally:
        if connection:
            connection.close()

if __name__ == "__main__":
    filename = r'P:\ETL\workshop_002\db_config.json'
    db_name = 'workshop2'
    table_name = 'grammy'

    df = extract_data(filename, db_name, table_name)

    if df is not None:
        print(df)
```

```python
    df = extract_data(filename, db_name, table_name)

    if df is not None:
        print(df)
```

```
      year                            title             published_at
0     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
1     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
2     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
3     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
4     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
...    ...                             ...                          ...
4805  1958   1st Annual GRAMMY Awards  (1958) 2017-11-28 08:03:45+00:00
4806  1958   1st Annual GRAMMY Awards  (1958) 2017-11-28 08:03:45+00:00
4807  1958   1st Annual GRAMMY Awards  (1958) 2017-11-28 08:03:45+00:00
4808  1958   1st Annual GRAMMY Awards  (1958) 2017-11-28 08:03:45+00:00
4809  1958   1st Annual GRAMMY Awards  (1958) 2017-11-28 08:03:45+00:00

                  updated_at  \
0    2020-05-19 12:10:28+00:00
1    2020-05-19 12:10:28+00:00
2    2020-05-19 12:10:28+00:00
3    2020-05-19 12:10:28+00:00
4    2020-05-19 12:10:28+00:00
...                        ...
4805 2019-09-10 08:11:09+00:00
4806 2019-09-10 08:11:09+00:00
4807 2019-09-10 08:11:09+00:00
4808 2019-09-10 08:11:09+00:00
4809 2019-09-10 08:11:09+00:00
...
4808                                        None    True
4809                                        None    True
```

We extract the data from database, is called like df.

```
df.head()
```

Python

	year	title	published_at	updated_at	category	nominee	artist	workers	img	winner
0	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19 12:10:28+00:00	2020-05-19 12:10:28+00:00	Record Of The Year	Bad Guy	Billie Eilish	Finneas O'Connell, producer; Rob Kinelski & Fi...	https://www.grammy.com/sites/com/files/styles/...	True
1	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19 12:10:28+00:00	2020-05-19 12:10:28+00:00	Record Of The Year	Hey, Ma	Bon Iver	BJ Burton, Brad Cook, Chris Messina & Justin V...	https://www.grammy.com/sites/com/files/styles/...	True
2	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19 12:10:28+00:00	2020-05-19 12:10:28+00:00	Record Of The Year	7 rings	Ariana Grande	Charles Anderson, Tommy Brown, Michael Foster ...	https://www.grammy.com/sites/com/files/styles/...	True
3	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19 12:10:28+00:00	2020-05-19 12:10:28+00:00	Record Of The Year	Hard Place	H.E.R.	Rodney "Darkchild" Jerkins, producer; Joseph H...	https://www.grammy.com/sites/com/files/styles/...	True
4	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19 12:10:28+00:00	2020-05-19 12:10:28+00:00	Record Of The Year	Talk	Khalid	Disclosure & Denis Kosiak, producers; Ingmar C...	https://www.grammy.com/sites/com/files/styles/...	True

```
df.shape
```

```
(4810, 10)
```

Our dataset has a dimension of (4810, 10), i.e. 4,810 grammy, and consists of 10 columns or atributes.

En los valores unicos tenemos

```
df.nunique()
```

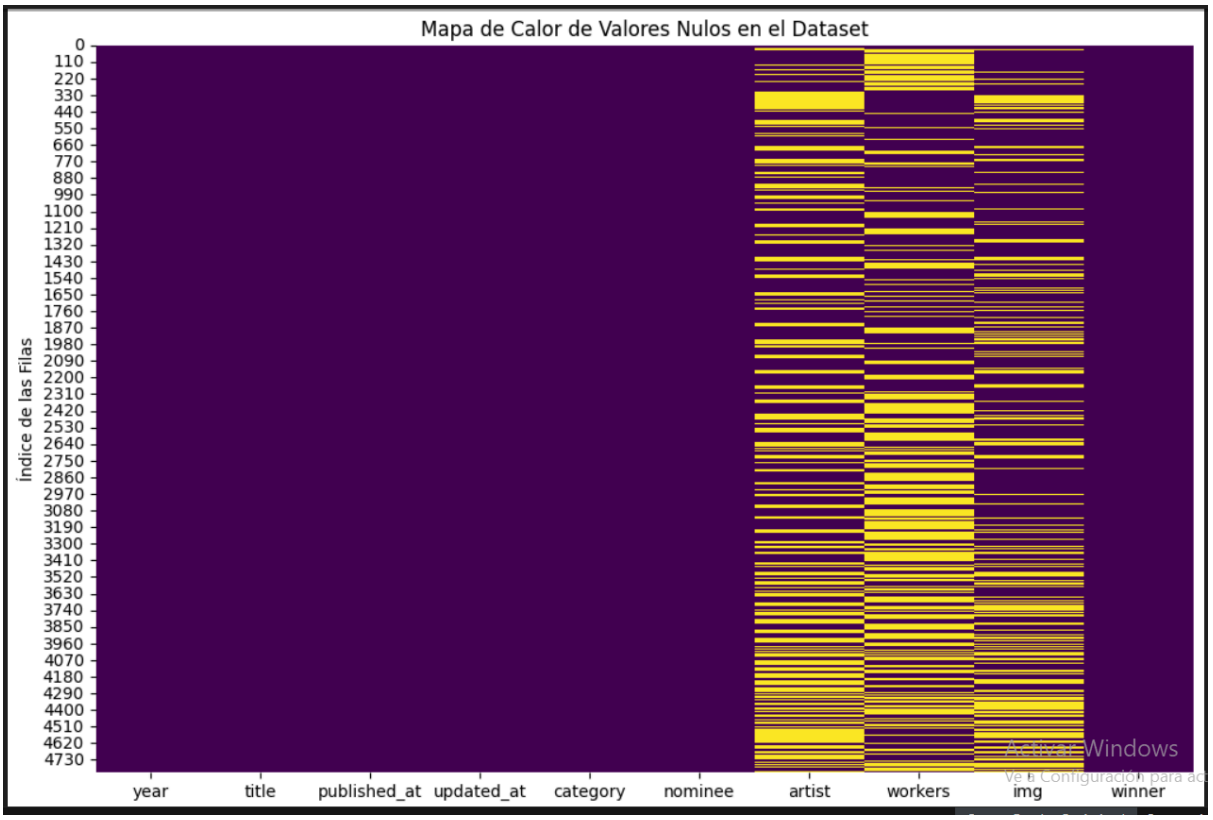
```
year            62
title           62
published_at     4
updated_at      10
category       638
nominee       4131
artist        1658
workers       2366
img           1463
winner           1
dtype: int64
```

This is telling us that in the winners column there is only one type of data and when we check it is "TRUE" then we can understand it as that all the records are those who won a grammy.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4810 entries, 0 to 4809
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   year          4810 non-null   int64
 1   title         4810 non-null   object
 2   published_at  4810 non-null   datetime64[ns, UTC]
 3   updated_at    4810 non-null   datetime64[ns, UTC]
 4   category      4810 non-null   object
 5   nominee       4804 non-null   object
 6   artist        2970 non-null   object
 7   workers       2620 non-null   object
 8   img           3443 non-null   object
 9   winner        4810 non-null   bool
```

Make a heat map to review nulls by column in a more graphical way




```
df.isnull().sum()
```

```
year              0
title             0
published_at      0
updated_at        0
category          0
nominee           6
artist         1840
workers        2190
img            1367
winner            0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

Here, we don't have duplicates and we can see that we are missing an artist(1840), workers(2190) and img(1367), let's review

```python
rows_with_nulls = df[df['artist'].isnull() & df['workers'].isnull()]
print(rows_with_nulls)
num_rows = (df['artist'].isnull() & df['workers'].isnull()).sum()
print("Número de registros con 'artist' y 'workers' nulos:", num_rows)
```

```
      year                          title             published_at  \
25    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
26    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
27    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
28    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
29    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
30    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
31    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
32    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
437   2018  61st Annual GRAMMY Awards  (2018) 2018-12-07 07:48:49+00:00
505   2018  61st Annual GRAMMY Awards  (2018) 2018-12-07 07:48:49+00:00
509   2018  61st Annual GRAMMY Awards  (2018) 2018-12-07 07:48:49+00:00
523   2017  60th Annual GRAMMY Awards  (2017) 2018-05-22 10:08:24+00:00
591   2017  60th Annual GRAMMY Awards  (2017) 2018-05-22 10:08:24+00:00
595   2017  60th Annual GRAMMY Awards  (2017) 2018-05-22 10:08:24+00:00
610   2016  59th Annual GRAMMY Awards  (2016) 2017-11-28 08:03:45+00:00
676   2016  59th Annual GRAMMY Awards  (2016) 2017-11-28 08:03:45+00:00
680   2016  59th Annual GRAMMY Awards  (2016) 2017-11-28 08:03:45+00:00
695   2015  58th Annual GRAMMY Awards  (2015) 2017-11-28 08:03:45+00:00
761   2015  58th Annual GRAMMY Awards  (2015) 2017-11-28 08:03:45+00:00
765   2015  58th Annual GRAMMY Awards  (2015) 2017-11-28 08:03:45+00:00
780   2014  57th Annual GRAMMY Awards  (2014) 2017-11-28 08:03:45+00:00
845   2014  57th Annual GRAMMY Awards  (2014) 2017-11-28 08:03:45+00:00
849   2014  57th Annual GRAMMY Awards  (2014) 2017-11-28 08:03:45+00:00
863   2013  56th Annual GRAMMY Awards  (2013) 2017-11-28 08:03:45+00:00
```

We will proceed to eliminate the rows that do not have information in artist or workers since these values are essential for our analysis.

```python
mask = df['artist'].isnull() & df['workers'].isnull()
indices_to_drop = df[mask].index
df.drop(indices_to_drop, inplace=True)

num_rows_remaining = df.shape[0]
print("Número de registros restantes en el DataFrame:", num_rows_remaining)
print(df)
```

```
Número de registros restantes en el DataFrame: 4624
      year                          title             published_at  \
0     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
1     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
2     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
3     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
4     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
5     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
6     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
7     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
8     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
9     2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
10    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
11    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
12    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
13    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
14    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
15    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
16    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
17    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
18    2013  56th Annual GRAMMY Awards  (2013) 2017-11-28 08:03:45+00:00
19    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
20    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
21    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
22    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
```

With the records without artists or workers, I will proceed to look for the rows where there is no edge, we can find them in workers and fill them.

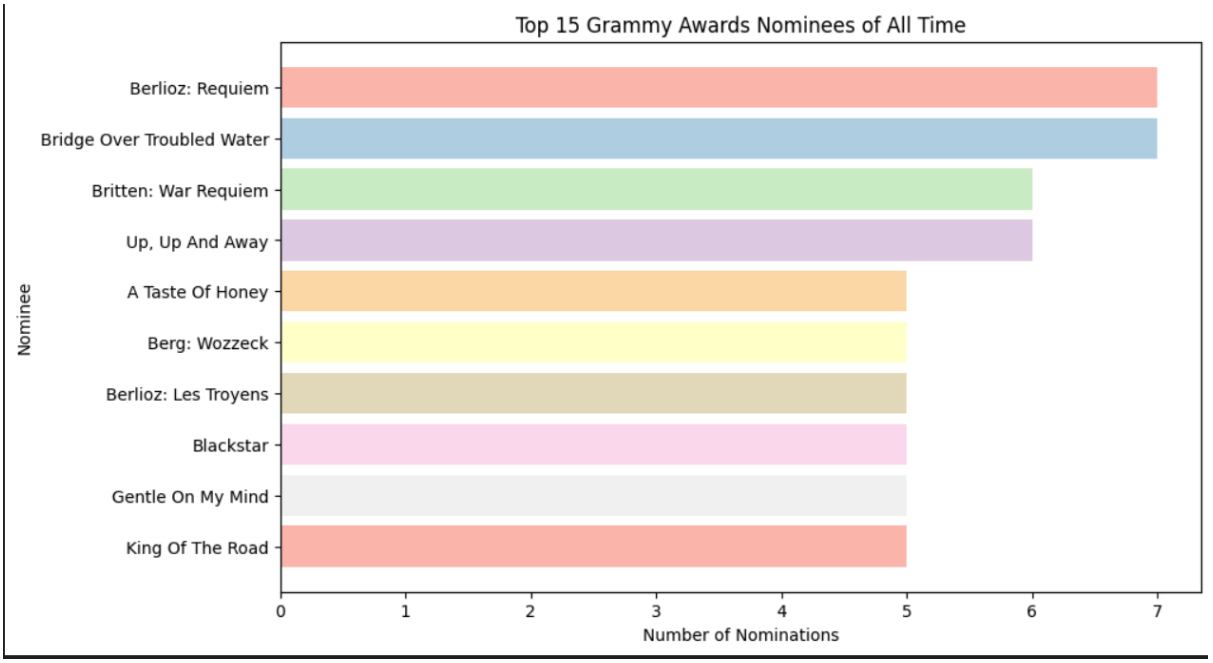
```python
mask = df['artist'].isnull() & df['workers'].notnull()

rows_with_workers_no_artist = df[mask]
print("Numero de registros donde 'artist' es nulo y 'workers' no es nulo:", len(rows_with_workers_no_artist))
print(rows_with_workers_no_artist)

pd.set_option('display.max_rows', None)
print(rows_with_workers_no_artist)
```

```
Numero de registros donde 'artist' es nulo y 'workers' no es nulo: 1654
      year                          title              published_at  \
16    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
17    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
19    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
20    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
21    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
22    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
23    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
24    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
78    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
79    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
80    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
81    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
82    2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
103   2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
104   2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
105   2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
106   2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
107   2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
128   2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
129   2019  62nd Annual GRAMMY Awards  (2019) 2020-05-19 12:10:28+00:00
```

```python
artists = df['artist'].value_counts().head()
print(artists)
```

```
artist
(Various Artists)    66
U2                   18
Aretha Franklin      16
Ella Fitzgerald      13
Bruce Springsteen    13
Name: count, dtype: int64
```

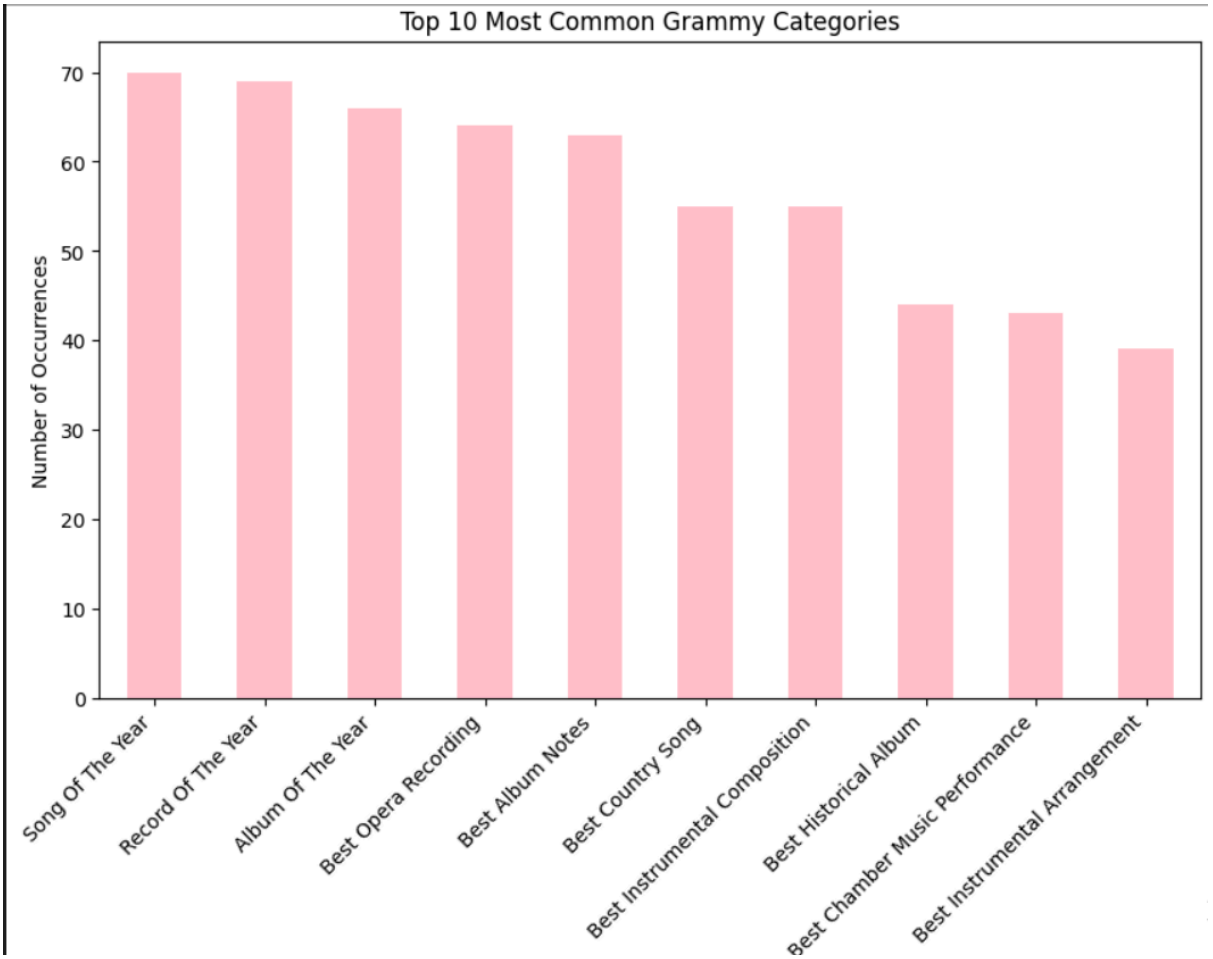
We have a total of 1654 records where we can find the artist by names in workers. This process will be done in Airflow transformations.



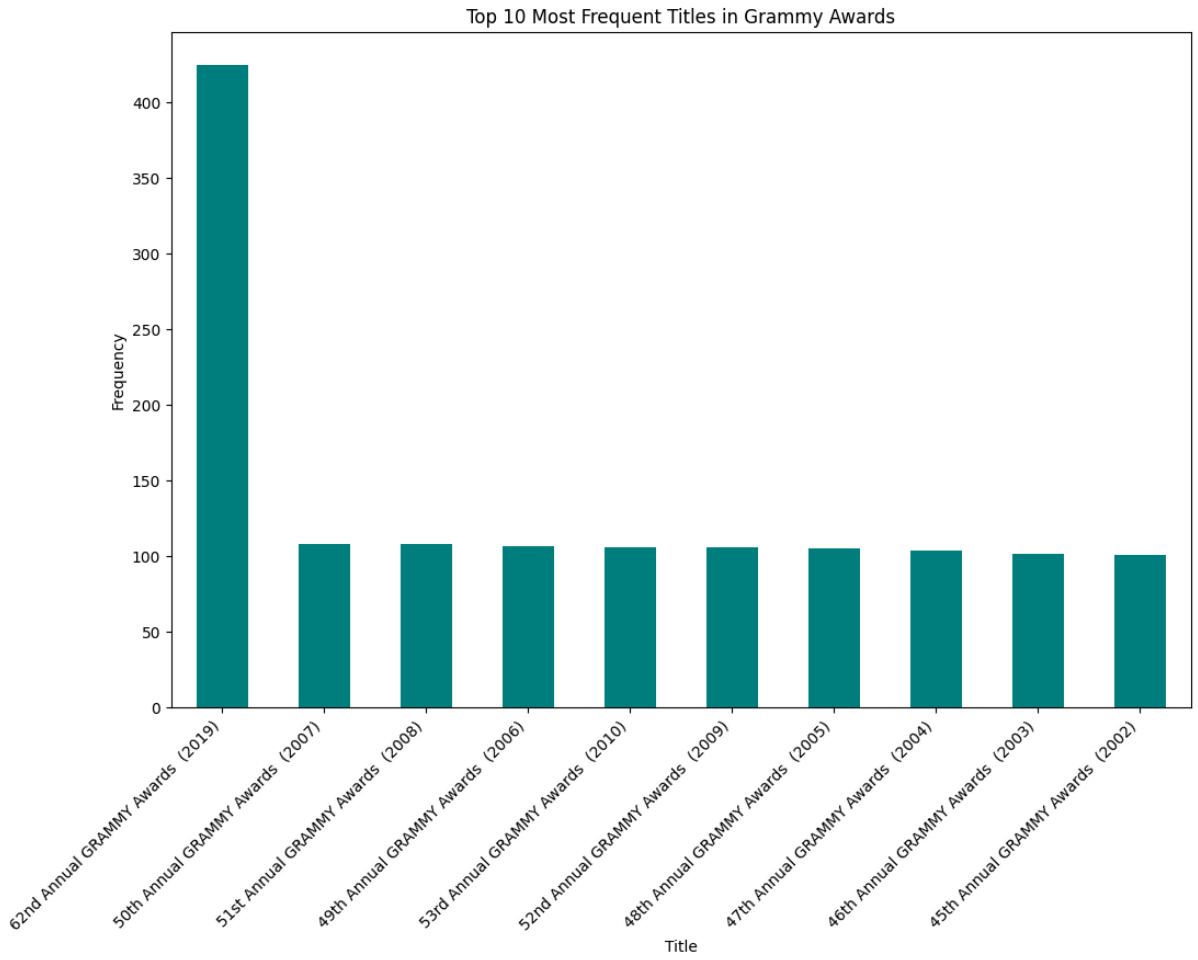
Here we can see a Top 15 Grammy winners from all the time they cover in the dataset

```
df['category'].value_counts()
```

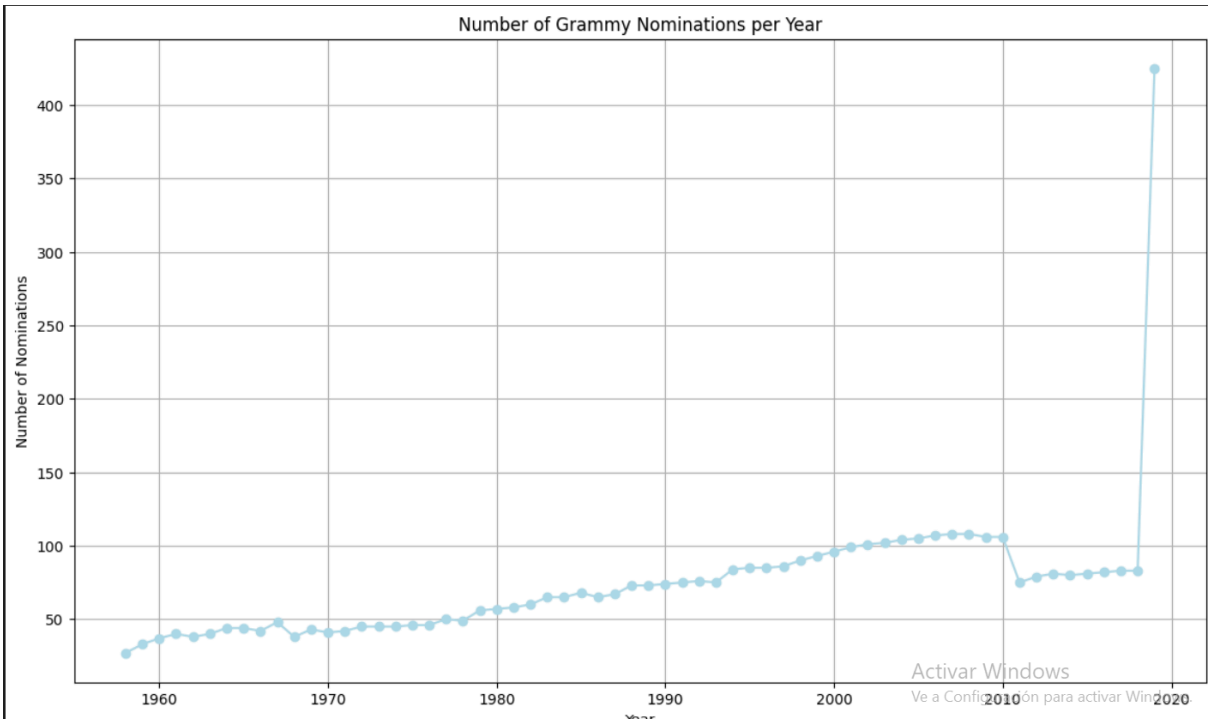
```
category
Song Of The Year                                                  70
Record Of The Year                                                69
Album Of The Year                                                 66
Best Opera Recording                                              64
Best Album Notes                                                  63
Best Country Song                                                 55
Best Instrumental Composition                                     55
Best Historical Album                                             44
Best Chamber Music Performance                                    43
Best Instrumental Arrangement                                     39
Best Country Instrumental Performance                             38
Best Orchestral Performance                                       36
Best Classical Album                                              35
Best Pop Instrumental Performance                                 34
Best Bluegrass Album                                              34
Best New Age Album                                                33
Best Rock Song                                                    33
Best Reggae Album                                                 33
Best Rhythm & Blues Song                                          32
Best Recording Package                                            32
Best Metal Performance                                            32
Best Recording For Children                                       31
Best Country Album                                                30
Best R&B Album                                                    30
...
Best Rhythm & Blues Solo Vocal Performance, Female                 1
Best Rhythm & Blues Solo Vocal Performance, Male                   1
Best Classical Performance - Instrumental Soloist (Without Orchestra)  1
Best Classical Performance - Operatic Or Choral                    1
```



Graph of the Top 10 most common categories



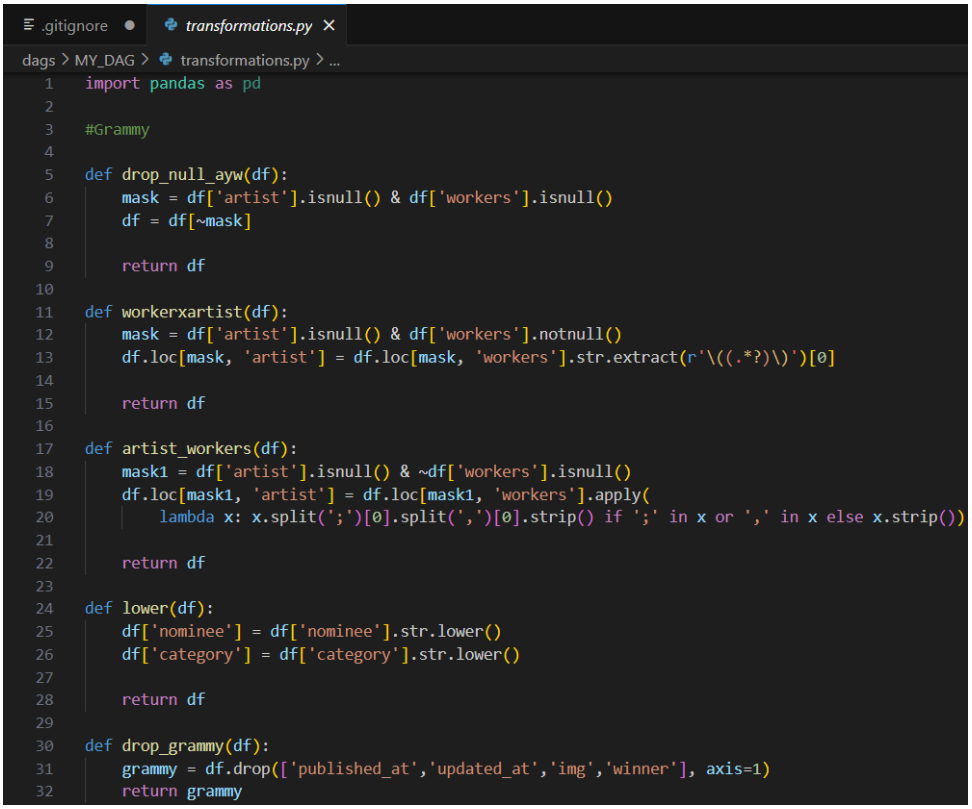
Graphs of the Top 10 names of the most frequent awards, here you can see that the 62nd Annual GRAMMY Awards in 2019 is very outstanding since in 2019 there are many more registrations than in the previous years, and we can see by looking at the other frequencies of the other names by year.



We can see that have more registers in 2019 than on other years, showing why 2019 is in the top 1 and the others have the same number of records.

Step 2: Airflow

We proceed to create the files to run our workshop in airflow.
1 archivo: Transformations.py



```python
import pandas as pd

#Grammy

def drop_null_ayw(df):
    mask = df['artist'].isnull() & df['workers'].isnull()
    df = df[~mask]

    return df

def workerxartist(df):
    mask = df['artist'].isnull() & df['workers'].notnull()
    df.loc[mask, 'artist'] = df.loc[mask, 'workers'].str.extract(r'\((.*?)\)')[0]

    return df

def artist_workers(df):
    mask1 = df['artist'].isnull() & ~df['workers'].isnull()
    df.loc[mask1, 'artist'] = df.loc[mask1, 'workers'].apply(
        lambda x: x.split(';')[0].split(',')[0].strip() if ';' in x or ',' in x else x.strip())

    return df

def lower(df):
    df['nominee'] = df['nominee'].str.lower()
    df['category'] = df['category'].str.lower()

    return df

def drop_grammy(df):
    grammy = df.drop(['published_at','updated_at','img','winner'], axis=1)
    return grammy
```

```python


def filter_songs(df):
    include_mask = (df['category'].str.contains('song', case=False, na=False) |
                    df['category'].str.contains('performance', case=False, na=False) |
                    df['category'].str.contains(r'\brecord\b', case=False, na=False, regex=True))

    exclude_mask = (df['category'].str.contains('album', case=False) |
                    df['category'].str.contains('artist', case=False))

    combined_mask = include_mask & ~exclude_mask

    return df[combined_mask]

```

In this file are all the modifications we made to Grammy's data for the process. We eliminate the rows where artist and workers are null, where artist is null but workers we do not take the first name in workers and we put it for artist, we put lower columns and we eliminate the columns that did not serve us for what we want to see in our dashboard.

```python
#Spotify      You, 5 days ago • avance

def drop_columns(df):
    columns_to_remove = ['Unnamed: 0', 'duration_ms', 'danceability', 'energy', 'key',
                         'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness',
                         'liveness', 'valence', 'tempo', 'time_signature']
    df.drop(columns=[col for col in columns_to_remove if col in df.columns], axis=1, inplace=True)

    return df

def drop_nulls(df):
    mask = df.isnull().any(axis=1)
    df = df[~mask]

    return df

def drop_duplicates1(df):
    df['track_name'] = df['track_name'].str.lower()
    indices_max_popularity = df.groupby(['track_id', 'track_name', 'artists'])['popularity'].idxmax()
    df = df.loc[indices_max_popularity]
    df.reset_index(drop=True, inplace=True)

    return df


def drop_duplicates2(df):
    df['track_name'] = df['track_name'].str.lower()
    spotify = df.loc[df.groupby(['track_name', 'artists'])['popularity'].idxmax()]

    return spotify
```

In this file are all the transformations that we reviewed and said to do on the Spotify data. We eliminate the columns that we will not use for our analysis in the Dashboard, the rows that are null because it would be of no use to us, the duplicates by id, name and artist and we leave the one that is most popular, we also do the same but now only to the duplicates by name and artist and we also leave the most popular.

2 Archivo: ETL.PY

In this file are all the functions calling the transformations that we performed previously, these functions are the ones that we will call in the next file in our dag for Airflow.

```python
import requests
import psycopg2
import pandas as pd
import json
import logging
import os
import transformations
```

```python
def extract_csv():
    csv = r"data/spotify_dataset.csv"
    df_spotify = pd.read_csv(csv)
    logging.info(f"Columns are: {df_spotify.columns}")

    jdata_spotify = df_spotify.to_json(orient='records')
    return jdata_spotify

def extract_basedatos():
    with open(r"db_config.json") as config_json:
        config = json.load(config_json)

    conx = psycopg2.connect(**config)
    try:
        mycursor = conx.cursor()

        all_info = "SELECT * from grammy"
        mycursor.execute(all_info)
        results = mycursor.fetchall()
        df_grammy = pd.DataFrame(results, columns=['id', 'year', 'title', 'published_at', 'updated_at', 'category', 'nominee', 'artist',

        logging.info(f"data is: {df_grammy.head()}")
        logging.info(f"Columns are: {df_grammy.columns}")

        jdata_grammy = df_grammy.to_json(orient='records')

        return jdata_grammy

    finally:
        mycursor.close()
        conx.close()
```

Here we can see the functions of extracting data from the spotify csv and from the database for grammy data.

```python
def transform_spotify(**kwargs):
    ti = kwargs["ti"]
    json_data = json.loads(ti.xcom_pull(task_ids="extract_csv_task"))
    print("Data coming from extract:", json_data)
    print("Data type is: ", type(json_data))

    df_spotify = pd.json_normalize(json_data)

    df_spotify = transformations.drop_columns(df_spotify)
    df_spotify = transformations.drop_nulls(df_spotify)
    df_spotify = transformations.drop_duplicates1(df_spotify)
    df_spotify = transformations.drop_duplicates2(df_spotify)


    logging.info(f"Data after transformation is: {df_spotify.head()}")
    logging.info(f"Columns after transformation are: {df_spotify.columns}")

    jdata_spotify = df_spotify.to_json(orient='records')

    return jdata_spotify
```

This is the function to make the transformations for the spotify data, it is called and the spotify df is returned

```python
def transform_grammy(**kwargs):
    ti = kwargs["ti"]
    json_data = json.loads(ti.xcom_pull(task_ids="extract_bd_task"))
    print("data coming from extract:", json_data)
    print("data type is: ", type(json_data))

    df_grammy = pd.json_normalize(json_data)

    df_grammy = transformations.drop_null_ayw(df_grammy)
    df_grammy = transformations.workerxartist(df_grammy)
    df_grammy = transformations.artist_workers(df_grammy)
    df_grammy = transformations.lower(df_grammy)
    df_grammy = transformations.drop_grammy(df_grammy)

    jdata_grammy = df_grammy.to_json(orient='records')

    return jdata_grammy
```

This is the function to make the transformations for the grammy data, it is called and the grammy df is returned

```python
def merge(**kwargs):
    ti = kwargs["ti"]
    json_data = json.loads(ti.xcom_pull(task_ids="transform_g_task"))
    df2 = pd.json_normalize(json_data)
    json_data = json.loads(ti.xcom_pull(task_ids="transform_s_task"))
    df1 = pd.json_normalize(json_data)

    logging.info("Data coming from Spotify extract:", df1)
    logging.info("Data type is:", type(df1))
    logging.info("Data coming from Grammy extract:", df2)
    logging.info("Data type is:", type(df2))

    df_spotify = pd.read_json(df1)
    df_grammys = pd.read_json(df2)

    df_merged = df_spotify.merge(df_grammys, how='left', left_on=['track_name', 'artists'], right_on=['nominee', 'artist'])
    df_merged.drop(columns=['nominee', 'artist'], inplace=True)

    jdata_merged = df_merged.to_json(orient='records')

    return jdata_merged
```

Here is the function to merge the already clean datasets according to our objective and what we wanted to achieve and analyze to visualize, then I perform a merge of these two datasets

by name of the songs and artist and nominee and artist, then I delete nominee and artist so that I don't have duplicates since these are the same

```python
def load(**kwargs):
    ti = kwargs["ti"]
    json_data = json.loads(ti.xcom_pull(task_ids="merge_task"))

    logging.info("Data coming from extract:", json_data)
    logging.info("Data type is:", type(json_data))

    df_grammys = pd.json_normalize(json_data)

    with open('db_config.json', 'r') as config_json:
        config = json.load(config_json)
    conx = psycopg2.connect(**config)

    try:
        mycursor = conx.cursor()

        mycursor.execute("""
            CREATE TABLE IF NOT EXISTS grammy_spotify_data (
                track_id VARCHAR(255) PRIMARY KEY,
                artists VARCHAR(255),
                album_name VARCHAR(255),
                track_name VARCHAR(255),
                popularity INT,
                explicit BOOLEAN,
                track_genre VARCHAR(255),
                year INT,
                title VARCHAR(255),
                category VARCHAR(255),
                workers VARCHAR(255)
            )
        """)

        for index, row in df_grammys.iterrows():
            values = [row['track_id'], row['artists'], row['album_name'], row['track_name'],
                      row['popularity'], row['explicit'], row['track_genre'], row['year'],
                      row['title'], row['category'], row['workers']]
```

```python
            workers VARCHAR(255)
        )
    """)

    for index, row in df_grammys.iterrows():
        values = [row['track_id'], row['artists'], row['album_name'], row['track_name'],
                  row['popularity'], row['explicit'], row['track_genre'], row['year'],
                  row['title'], row['category'], row['workers']]
        values = [None if pd.isna(value) else value for value in values]
        query = """
            INSERT INTO grammy_spotify_data (track_id, artists, album_name, track_name, popularity, explicit,
                                             track_genre, year, title, category, workers)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
        """
        mycursor.execute(query, values)

    conx.commit()
    logging.info("Data has been successfully loaded to the database.")

except Exception as e:
    logging.error("An error occurred:", e)
    conx.rollback()

finally:
    mycursor.close()
    conx.close()

df_grammys.to_csv("./data/grammy_spotify_data.csv")
return df_grammys.to_json(orient='records')
```

It is in the function to load the merge dataset to the database.

```python
def store(**kwargs):
    ti = kwargs["ti"]
    json_data = json.loads(ti.xcom_pull(task_ids="load_task"))
    df = pd.json_normalize(json_data)
    print("data coming from extract:", json_data)
    print("data type is: ", type(json_data))

    logging.info(f"data is {json_data}")

    #upload_csv("df_grammy.csv","1gq1Ih6mCI2_EgKDh5yV2LUKSqap9x2v6")
    logging.info( f"completed")
```

This is the function to load the merge to drive but it didn't work for me in the end.

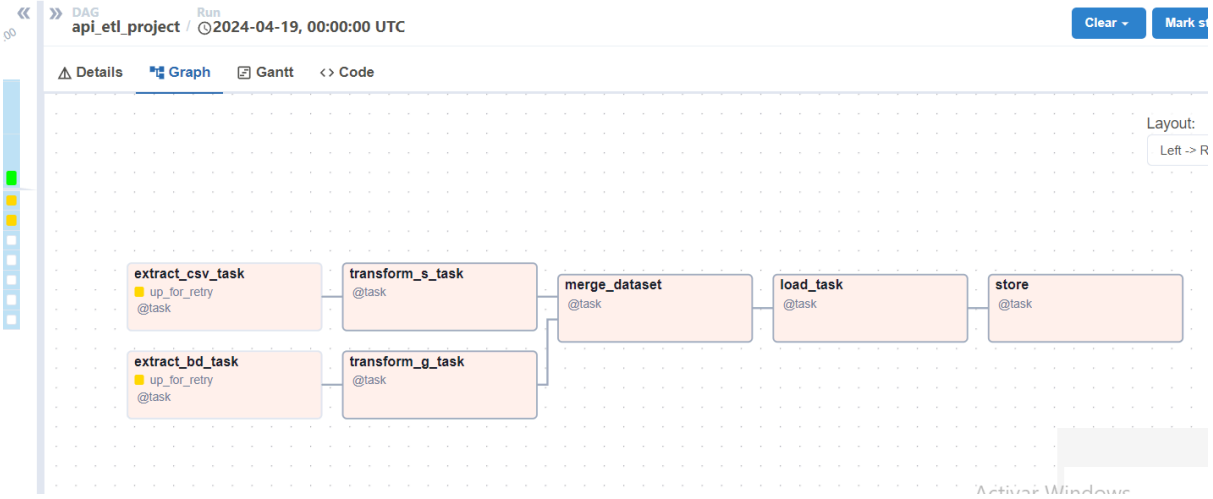
```python
extract_csv_task = PythonOperator(
    task_id = 'extract_csv_task',
    python_callable = extract_csv,
    provide_context = True,
)

transform_s_task = PythonOperator(
    task_id = 'transform_s_task',
    python_callable = transform_spotify,
    provide_context = True,
)

extract_bd_task = PythonOperator(
    task_id = 'extract_bd_task',
    python_callable = extract_basedatos,
    provide_context = True,
)

transform_g_task = PythonOperator(
    task_id = 'transform_g_task',
    python_callable = transform_grammy,
    provide_context = True,
)

store_task = PythonOperator(
    task_id='store_task',
    python_callable = store,
    provide_context = True,
)

load_task = PythonOperator(
    task_id ='load_task',
    python_callable = load,
    provide_context = True,
)
```

```python
extract_csv_task >> transform_s_task >> merge_task
extract_bd_task >> transform_g_task >> merge_task
merge_task >> load_task >> store_task
```

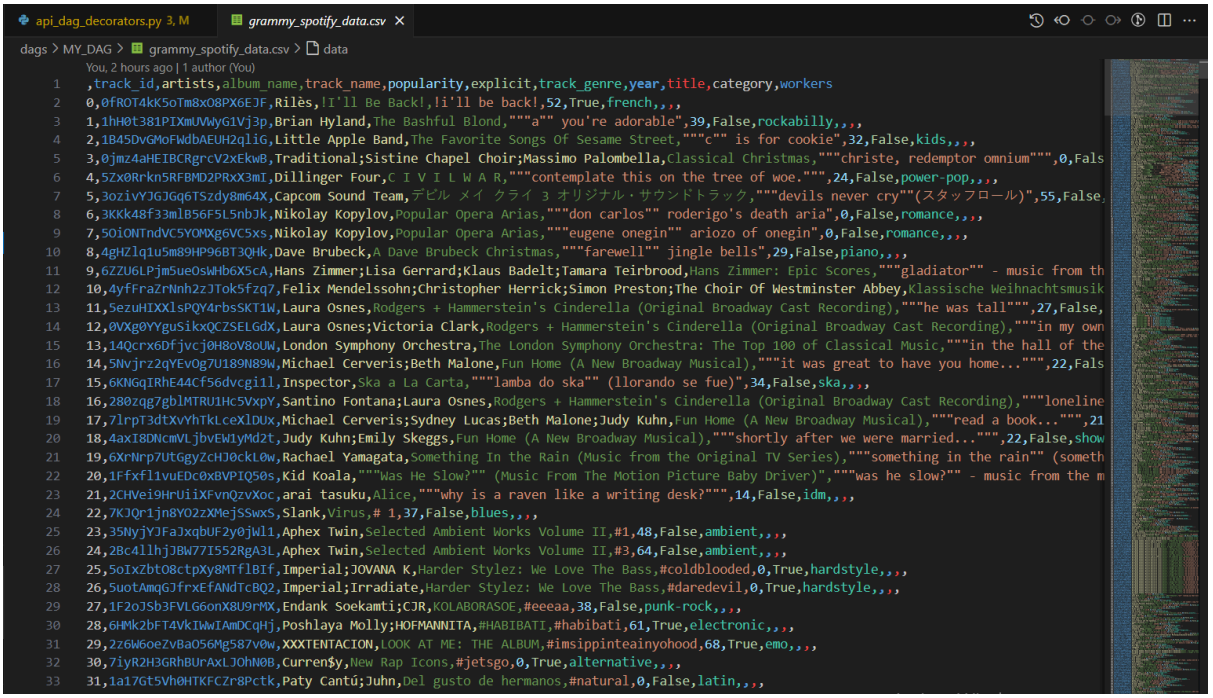
These are some of my dag's tasks for airflow.

3. Airflow

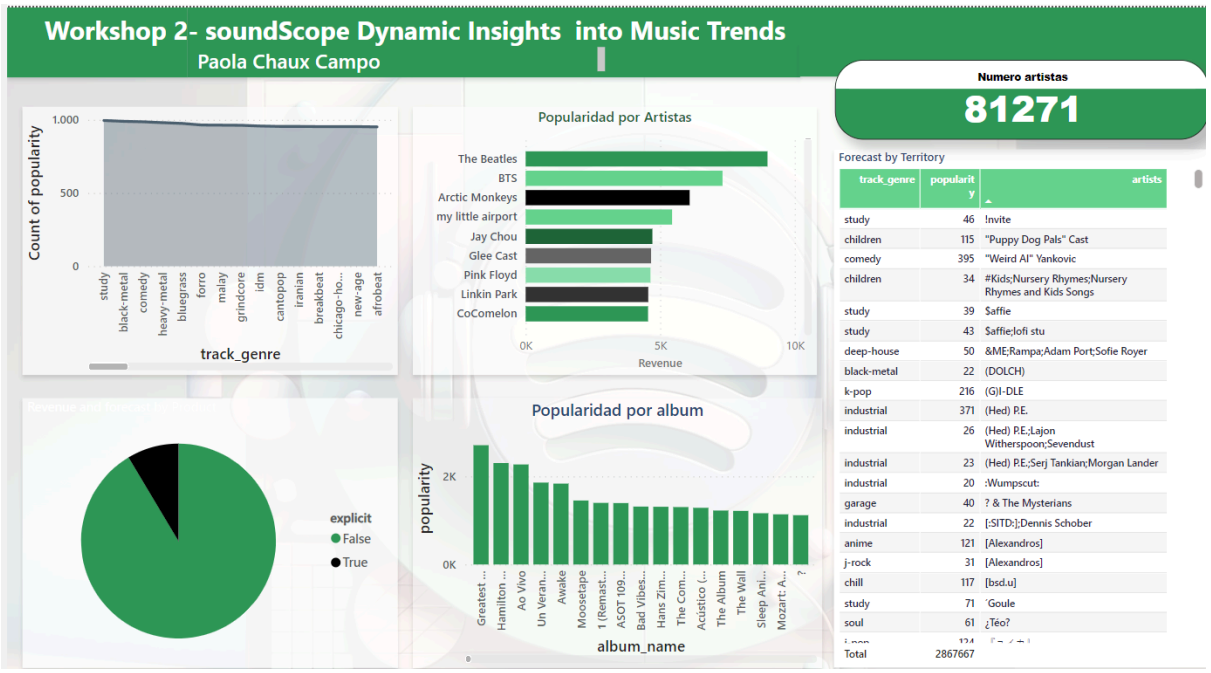
In Airflow the workflow is this:



merge



4. Dashboard



In this dashboard what was done was:

1. Frequency by genre, the first 3 are music for studying, black-metal and comedy.
2. Pie chart with popularity by songs look how many are explicit that there are very few.
3. Bar graph for popularity by artists where the most popular are The Beatles, BTS and Artic monkeys as the first 3.
4. Bar graph of popularity by album.
5. a table to see more versatile data by genre, popularity and artist.
6. The number of artists who have won a Grammy is 81,271.