Workshop #2 - Documentation

Martín García Chagüezá

About the workshop

In this workshop we will use two datasets (*spotify_dataset* and *the_grammys_awards*) that will be processed through Apache Airflow applying data cleaning, transformation and loading and storage, including a merge of both datasets. The result will culminate in visualizations on a dashboard that will give us important conclusions about this dataset.

The tools used are:

- Python 3.10 ➜ Download site
- Jupyter Notebook ➜ VS Code tool for using notebooks
- PostgreSQL ➜ Download site
- Power BI (Desktop version) ➜ Download site

The dependencies needed for Python are

- Apache Airflow
- Dotenv
- Pandas
- Matplotlib
- Seaborn
- SQLAlchemy
- PyDrive2

⚠ Apache Airflow **only runs** correctly in **Linux environments**. If you have Windows, we recommend using a virtual machine or WSL.

These dependencies are included in the `requirements.txt` file of the Python project. The step-by-step installation is described in the README file.

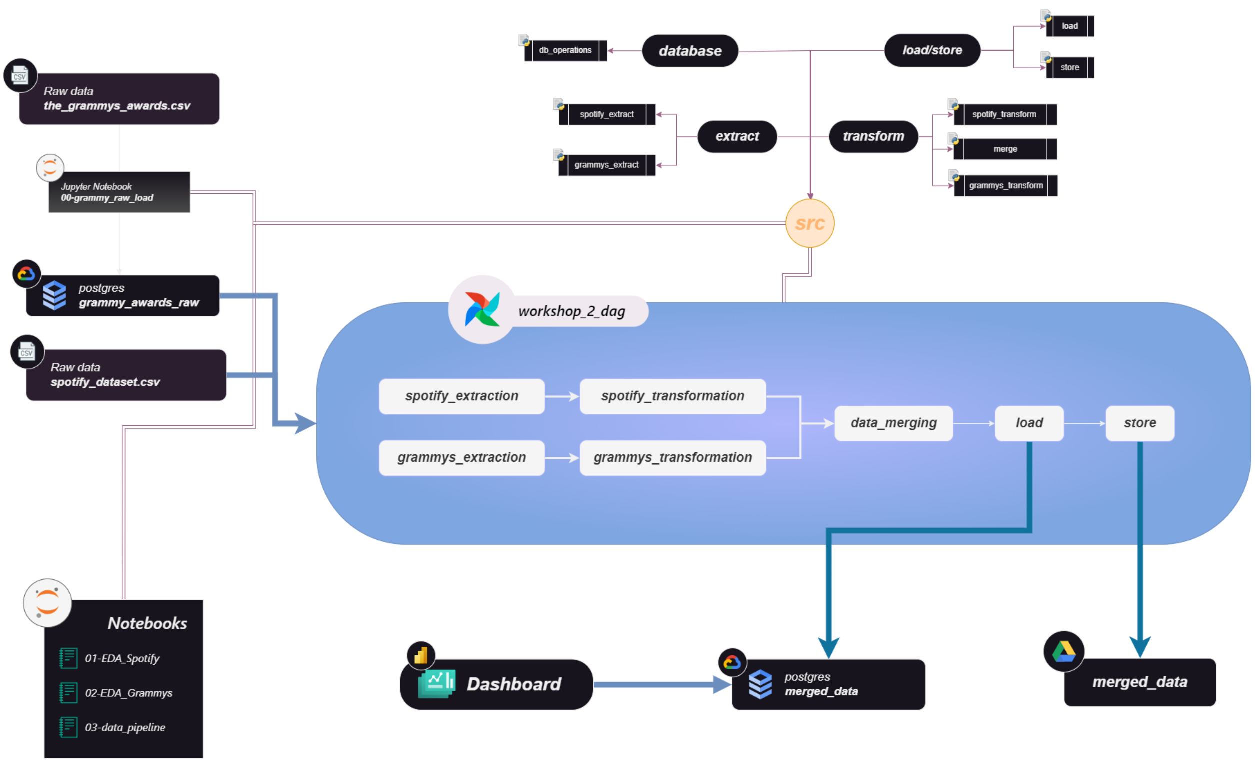
Goals of the project

Obtain the clean merged dataset for the creation of an analytical report using BI tools such as Power BI, which will be connected through a PostgreSQL database that will contain the transformed data set.

The analytical report will contain visualizations such as:

- **Nominated Songs per Genre**: A bar chart showing the number of nominated songs across different genres like Electronic/Dance, Rock/Metal, and Jazz and Soul.
- **Nominations per Artist**: A chart that displays how many nominations various artists, such as BTS, Adele, and Glee Cast, have received.
- **Energy Average per Track Genre**: A bar graph showing the average energy per track for different genres, indicating the relative energy levels of songs in each genre.
- **Danceability Average per Track Mood**: A chart showing the average danceability based on the mood of the track (e.g., Happy, Neutral, Sad).

Data flow



Process

Loading raw Grammys dataset

> ***Files used** → db_operations.py / 00-grammy_raw_load.ipynb*

Before we can run the EDA corresponding to the Grammy Awards dataset, we must upload it to a PostgreSQL database. **This process is mandatory if you wish to continue with the execution of the project.**

The process is described step by step in notebook 00, therefore, we are going to dwell on the details of the module in charge of the operations performed on the database.

db_operations.py - The details behind the DB connection and the load of the raw data

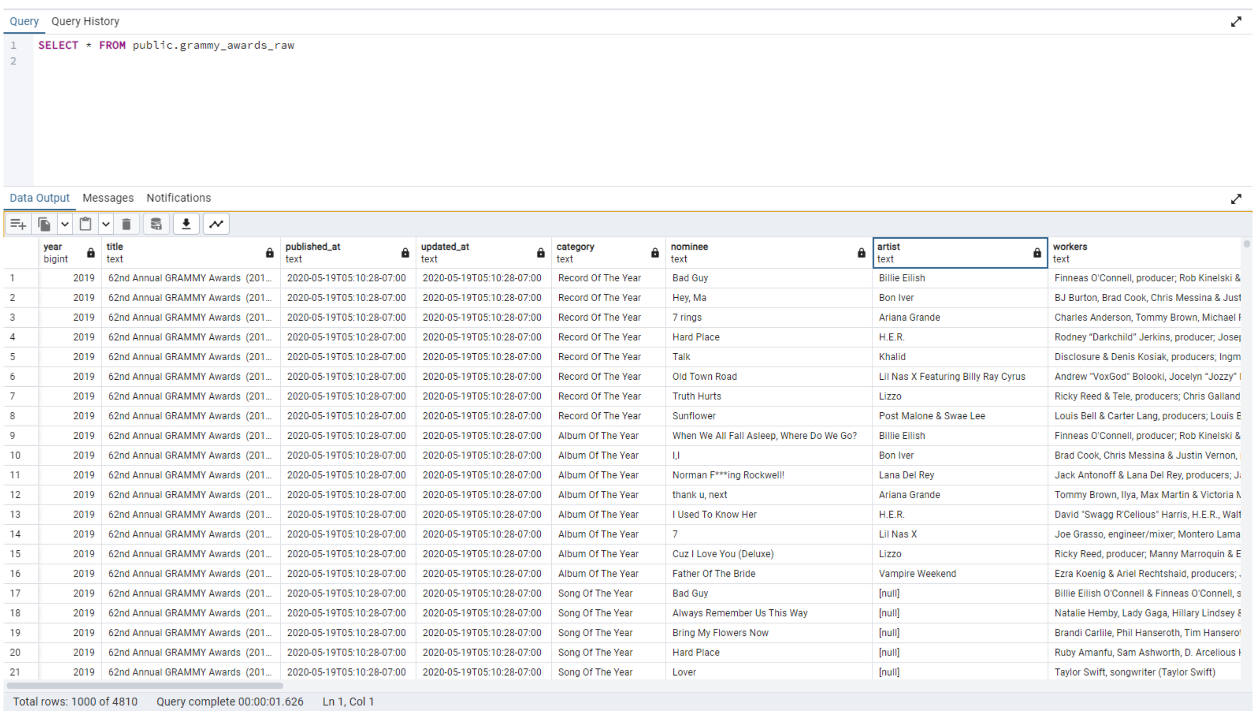
To run the connection engine we must specify a URL with the credentials of our database. These credentials are collected in the `.env` file that specifies the environment variables for our project.

Once we have these credentials, the engine is created (*as well as the database, in case it does not exist*); when we no longer need the engine, we have a function to terminate it and, thus, free memory.

```python
# Reading the environment variables
load_dotenv("../env/.env")

user = os.getenv("PG_USER")
password = os.getenv("PG_PASSWORD")

host = os.getenv("PG_HOST")
port = os.getenv("PG_PORT")

database = os.getenv("PG_DATABASE")

# Creating the connection engine from the URL made up of the environment variables
def creating_engine():
    url = f"postgresql://{user}:{password}@{host}:{port}/{database}"
    engine = create_engine(url)

    if not database_exists(url):
        create_database(url)
        logging.info("Database created")

    logging.info("Engine created. You can now connect to the database.")

    return engine

def disposing_engine(engine):
    engine.dispose()
    logging.info("Engine disposed.")
```

The loading of the raw data is specified in a function that receives the engine, the Pandas dataframe and the table name. The data types are inferred by Pandas through its `to_sql` function, which is in charge of transferring the data to the database.

```python
# Creating table and loading the raw data
def load_raw_data(engine, df, table_name):

    logging.info(f"Creating table {table_name} from Pandas DataFrame.")

    try:
        df.to_sql(table_name, con=engine, if_exists="replace", index=False)

        logging.info(f"Table {table_name} created successfully.")

    except Exception as e:
        logging.error(f"Error creating table {table_name}: {e}")
```



	year bigint	title text	published_at text	updated_at text	category text	nominee text	artist text	workers text
1	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Bad Guy	Billie Eilish	Finneas O'Connell, producer; Rob Kinelski &
2	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Hey, Ma	Bon Iver	BJ Burton, Brad Cook, Chris Messina & Just
3	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	7 rings	Ariana Grande	Charles Anderson, Tommy Brown, Michael
4	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Hard Place	H.E.R.	Rodney "Darkchild" Jerkins, producer; Jose
5	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Talk	Khalid	Disclosure & Denis Kosiak, producers; Ingm
6	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Old Town Road	Lil Nas X Featuring Billy Ray Cyrus	Andrew "VoxGod" Bolooki, Jocelyn "Jozzy"
7	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Truth Hurts	Lizzo	Ricky Reed & Tele, producers; Chris Galland
8	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Sunflower	Post Malone & Swae Lee	Louis Bell & Carter Lang, producers; Louis
9	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	When We All Fall Asleep, Where Do We Go?	Billie Eilish	Finneas O'Connell, producer; Rob Kinelski &
10	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	I,I	Bon Iver	Brad Cook, Chris Messina & Justin Vernon,
11	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	Norman F***ing Rockwell!	Lana Del Rey	Jack Antonoff & Lana Del Rey, producers; J
12	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	thank u, next	Ariana Grande	Tommy Brown, Ilya, Max Martin & Victoria
13	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	I Used To Know Her	H.E.R.	David "Swagg R'Celious" Harris, H.E.R., Walt
14	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	7	Lil Nas X	Joe Grasso, engineer/mixer; Montero Lama
15	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	Cuz I Love You (Deluxe)	Lizzo	Ricky Reed, producer; Manny Marroquin & E
16	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Album Of The Year	Father Of The Bride	Vampire Weekend	Ezra Koenig & Ariel Rechtshaid, producers;
17	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Song Of The Year	Bad Guy	[null]	Billie Eilish O'Connell & Finneas O'Connell, s
18	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Song Of The Year	Always Remember Us This Way	[null]	Natalie Hemby, Lady Gaga, Hillary Lindsey &
19	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Song Of The Year	Bring My Flowers Now	[null]	Brandi Carlile, Phil Hanseroth, Tim Hansero
20	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Song Of The Year	Hard Place	[null]	Ruby Amanfu, Sam Ashworth, D. Arcelious
21	2019	62nd Annual GRAMMY Awards (201...	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Song Of The Year	Lover	[null]	Taylor Swift, songwriter (Taylor Swift)

Total rows: 1000 of 4810 Query complete 00:00:01.626 Ln 1, Col 1

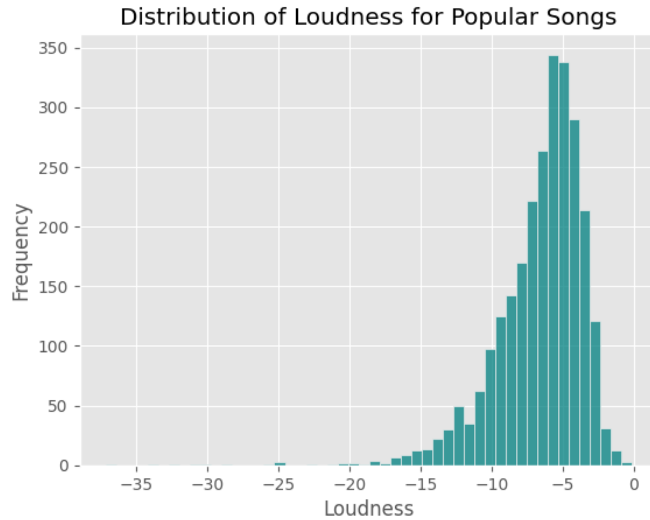
Exploring the Data - Spotify dataset

> ***Files used*** → *01-EDA_Spotify.ipynb*

In order not to make the analysis of certain variables more difficult, we chose to select a sample of the dataset: *the most popular songs* were selected from it. Some of the findings were:

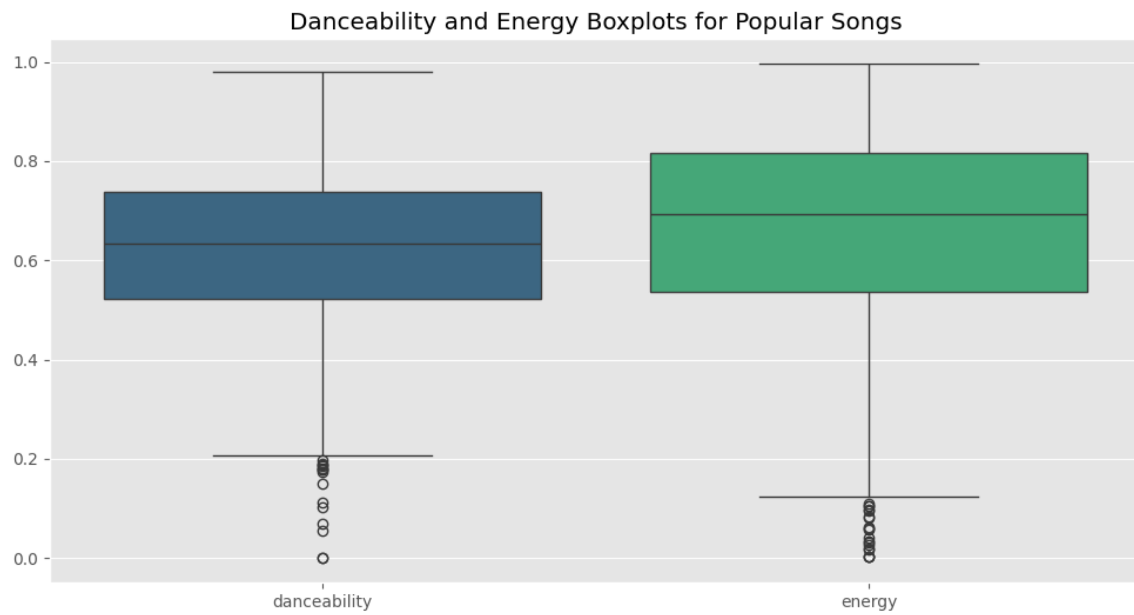
The most popular songs tend to have high volume levels

These high volume levels range from -10 dB to -2.5 dB, suggesting that the trend in popular music is to **maintain high volume levels**.



Popular songs show high variability in terms of `danceability` and `energy`

- For **energy**, the median of 0.7 implies that popular songs are typically energetic. This aligns with the idea that high-energy tracks are more likely to engage listeners and perform well on charts.
- Regarding **danceability**, the median of 0.65 suggests that popular songs generally have a moderate to high danceability. This indicates that the music industry tends to favor tracks that are easy to dance to, potentially to appeal to a wider audience.

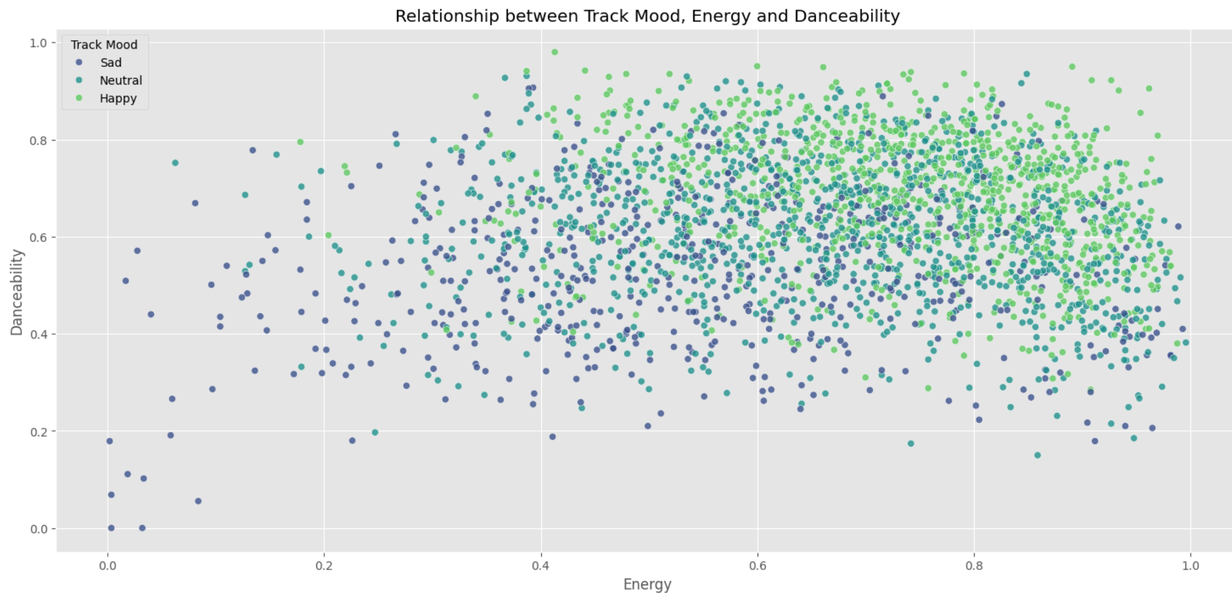


Happy songs tend to be more energetic and danceable

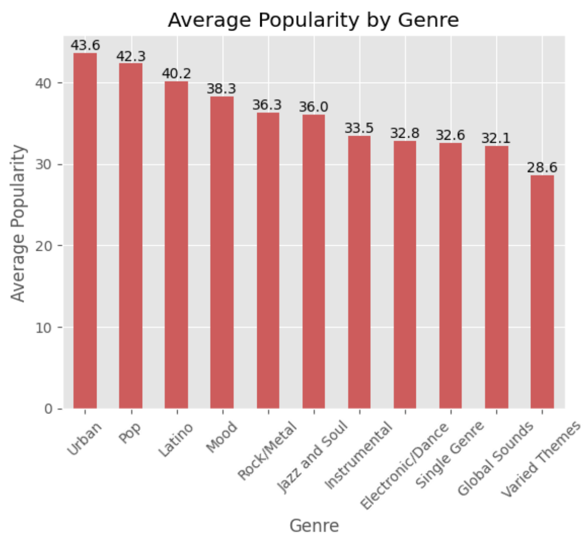
A scatter plot analysis reveals interesting patterns in the relationship between song moods and their energy and danceability characteristics:

- *Sad songs* tend to have **lower energy and danceability**.
- *Happy songs* generally cluster at **high levels of energy and danceability**.
- *Neutral songs* show a **more varied distribution**.

Regardless of mood, many popular songs tend to have moderate to high levels of energy and danceability, suggesting a general preference for these characteristics in mainstream music.



Urban, *Pop* and *Latin* have the highest average popularity



This suggests that the most produced genres **are not always the most popular**, while categories with fewer songs, such as *Urban*, have **a greater impact** on the audience.

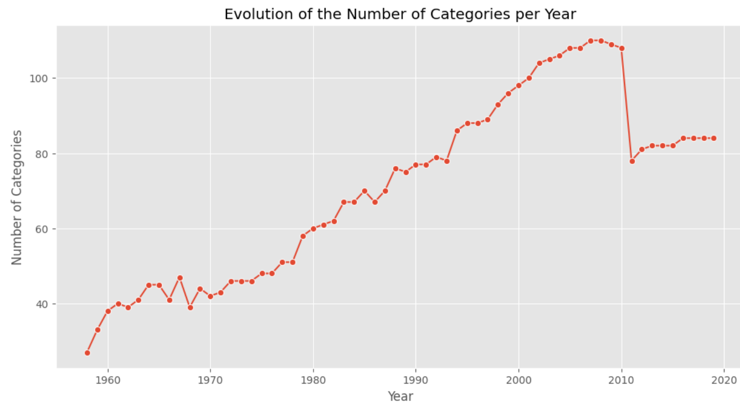
Exploring the Data - Grammys dataset

> ***Files used*** **→** → *db_operations.py / 02-EDA_Grammys.ipynb*

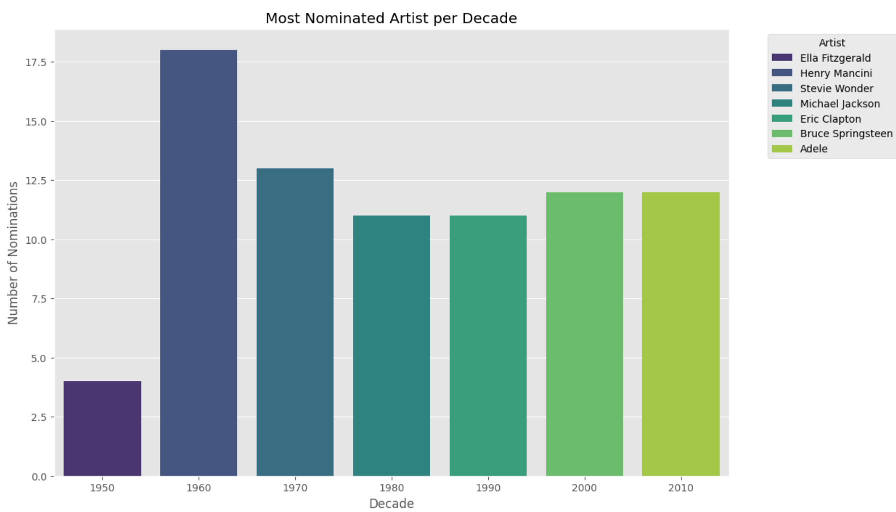
In order not to make the analysis of certain variables more difficult, we chose to select a sample of the dataset: *the most popular songs* were selected from it. Some of the findings were:

The number of Grammy nominations was steadily increased over the decades, but...

The number of Grammy nominations were at its peak in 2010, but a significant reorganization in 2011-2012 to streamline the awards caused a massive reduction in the number of nominations.



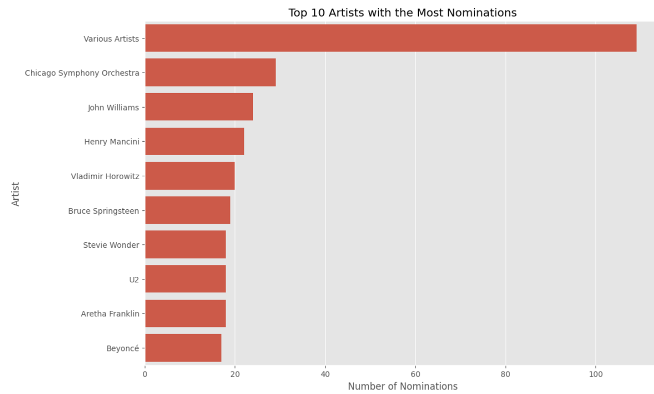
Certain artists dominated the nominations in specific decades



These trends correspond to changes in musical trends and popularity.

Collaborative or compilation albums received the highest number of nominations

These kind of nominations are represented by the title of *"Various Artists"*: they acummulate almost 100 nominations.



The structure of the Data Pipeline

> ***Files used*** *→ every module on* ***src*** */ 03-data_pipeline.ipynb*

This notebook goes through the process performed in Apache Airflow applying the functions of the different modules that are part of the ***src*** directory. However, special emphasis is placed on **the merge between the two datasets**.

Some of the remarks that can be highlighted are:

Extraction of the data

Here it's used both modules of the `src.extract` package. While the Spotify dataset is extracted locally, the Grammy Awards dataset is extracted by connecting to the database.

In [3]:

```
spotify_data = extracting_spotify_data("../data/spotify_dataset.csv")
```

```
23/09/2024 09:11:49 PM Data extracted from ../data/spotify_dataset.csv.
```

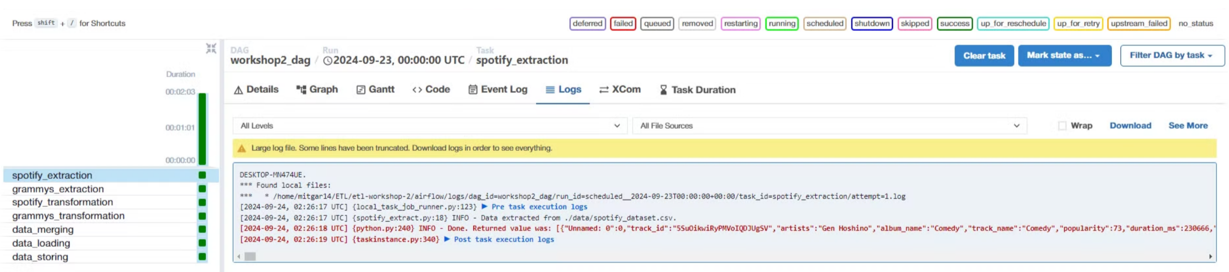
In [4]:

```
spotify_data.head()
```

Out[4]:

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	ene
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4

5 rows × 21 columns



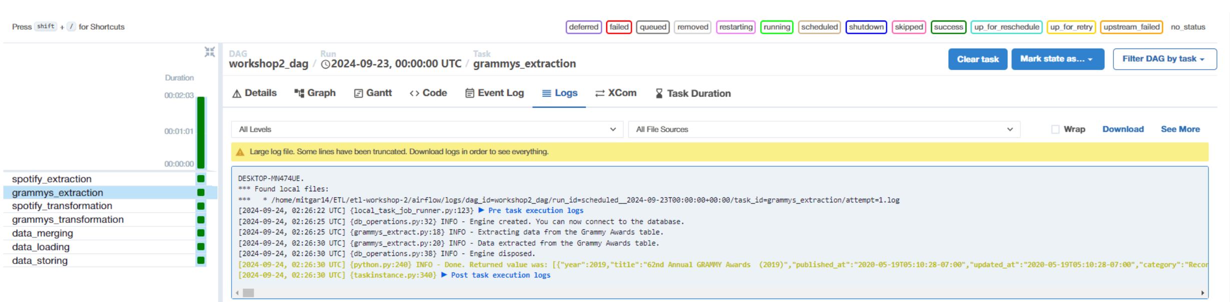
In [5]:
```
grammys_data = extracting_grammys_data()
```

```
23/09/2024 09:11:52 PM Engine created. You can now connect to the database.
23/09/2024 09:11:52 PM Extracting data from the Grammy Awards table.
23/09/2024 09:11:56 PM Data extracted from the Grammy Awards table.
23/09/2024 09:11:56 PM Engine disposed.
```

In [6]:
```
grammys_data.head()
```

Out[6]:

	year	title	published_at	updated_at	category	nominee	artist	workers	img	wi
0	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Bad Guy	Billie Eilish	Finneas O'Connell, producer; Rob Kinelski & Fi...	https://www.grammy.com/sites/com/files/styles/...	
1	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Hey, Ma	Bon Iver	BJ Burton, Brad Cook, Chris Messina & Justin V...	https://www.grammy.com/sites/com/files/styles/...	
2	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	7 rings	Ariana Grande	Charles Anderson, Tommy Brown, Michael Foster ...	https://www.grammy.com/sites/com/files/styles/...	
3	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Hard Place	H.E.R.	Rodney "Darkchild" Jerkins, producer; Joseph H...	https://www.grammy.com/sites/com/files/styles/...	
4	2019	62nd Annual GRAMMY Awards (2019)	2020-05-19T05:10:28-07:00	2020-05-19T05:10:28-07:00	Record Of The Year	Talk	Khalid	Disclosure & Denis Kosiak, producers; Ingmar C...	https://www.grammy.com/sites/com/files/styles/...	

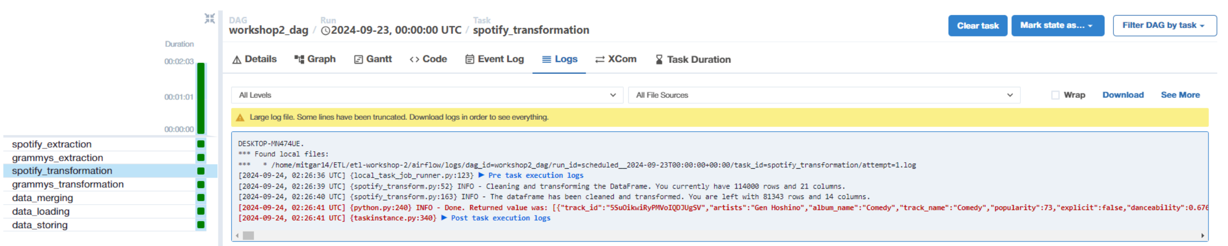


Transformations in the *Spotify* dataset

> ***module used was →*** `transform.spotify_transform`

- Removing unnecessary columns (e.g., `"Unnamed: 0"`).
- Eliminating null values and resetting the DataFrame index.

- Removing duplicates through several steps:
  - Dropped exact duplicate rows.
  - Removed duplicates based on the `"track_id"` column.
  - Mapped detailed genres to broader categories using a predefined genre mapping dictionary.
  - Dropped duplicates based on song names and artists, keeping the most popular entries.
- Generated new columns for enhanced data analysis:
  - **`duration_min`**: Converted song duration from milliseconds to minutes.
  - **`duration_category`**: Categorized songs based on their duration.
  - **`popularity_category`**: Categorized songs based on their popularity scores.
  - **`track_mood`**: Identified the mood of songs using valence scores.
  - **`live_performance`**: Flagged songs with a high likelihood of being live performances.
- Dropped irrelevant columns to streamline the dataset (e.g., `"loudness"`, `"mode"`, `"tempo"`).

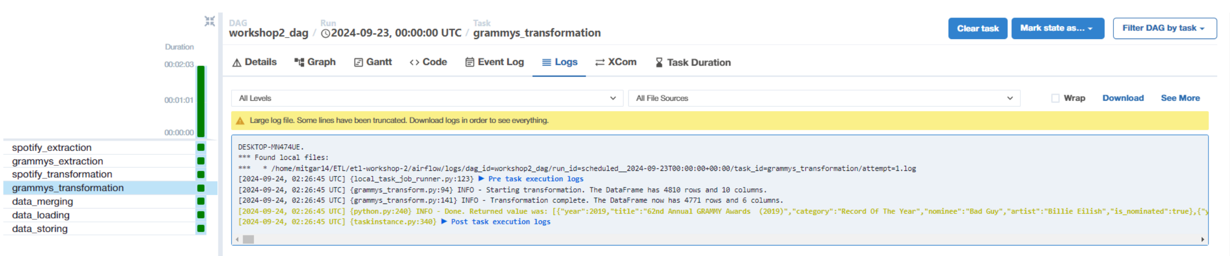


Transformations in the *Grammys* dataset

> ***module used was →*** `transform.grammys_transform`

- Renaming the column `winner` to `is_nominated`.
- Dropping unnecessary columns (e.g., `published_at`, `updated_at`, `img`).

- Removing rows with null values in `nominee`.
- Handling cases where both `artist` and `workers` are null:
  - Filtered out specific categories listed in the `categories` list.
  - For the remaining rows, filled `artist` with the value from `nominee`.
- Populating the `artist` column by applying several functions:
  - **`extract_artist`**: Extracted artist names within parentheses from the `workers` column.
  - **`move_workers_to_artist`**: Moved data from `workers` to `artist` if `artist` is null and `workers` doesn't contain semicolons or commas.
  - **`extract_artists_before_semicolon`**: Extracted artist names before semicolons in `workers`, excluding any roles of interest.
  - **`extract_roles_based_on_interest`**: Extracted names associated with specific roles defined in the `roles_of_interest` list from `workers`.
- Dropped rows with null values in `artist`.
- Replaced certain values in the `artist` column (e.g., changing `(Various Artists)` to `Various Artists`).
- Dropped the `workers` column as it was no longer needed.



Merging the datasets

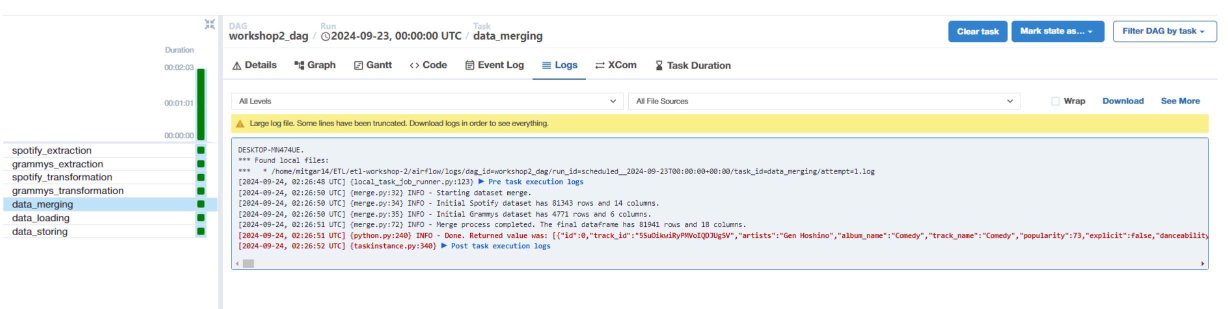
> ***module used was →*** `transform.merge`

Before listing the processes that occurred within the module it should be noted that the criteria chosen for the merge, although effective for most cases, yields quite a few false positives.

In order to fix these errors we must include the artist's name in the merge criteria. However, it would be necessary to analyze artist one by one if we want a successful merge.

The cleaning process executed was:

- **Cleaning key columns for accurate merging**:
  - Converted the `track_name` column in the Spotify DataFrame to lowercase and stripped whitespace, creating a new column `track_name_clean`.
  - Converted the `nominee` column in the Grammys DataFrame to lowercase and stripped whitespace, creating a new column `nominee_clean`.
- **Merging the datasets**:
  - Performed a left join on the cleaned columns `track_name_clean` and `nominee_clean` to merge the DataFrames.
  - Used suffixes to differentiate overlapping columns, appending `_grammys` to columns from the Grammys DataFrame when necessary.
- **Handling missing values**:
  - Filled null values in the `title` and `category` columns with `"Not applicable"`.
  - Filled null values in the `is_nominated` column with `False`.
- **Dropping unnecessary columns**:
  - Removed columns that were no longer needed after the merge, such as `"year"`, `"artist"`, `"nominee"`, `"nominee_clean"`, and `"track_name_clean"`.



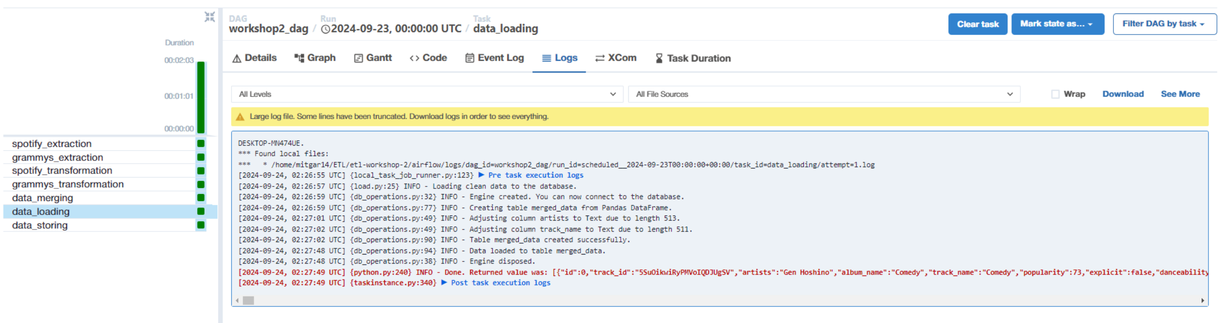
Loading the data

Although it is not necessary for the notebook, it is recommended that the data be uploaded to a cloud database due to the complexity of handling PostgreSQL in WSL, especially when handling Apache Airflow.

In [18]:

```python
loading_merged_data(merged_df, "merged_data")
```

```
23/09/2024 09:11:58 PM Loading clean data to the database.
23/09/2024 09:12:01 PM Engine created. You can now connect to the database.
23/09/2024 09:12:01 PM Creating table merged_data from Pandas DataFrame.
23/09/2024 09:12:03 PM Adjusting column artists to Text due to length 513.
23/09/2024 09:12:03 PM Adjusting column track_name to Text due to length 511.
23/09/2024 09:12:04 PM Table merged_data created successfully.
23/09/2024 09:12:55 PM Data loaded to table merged_data.
23/09/2024 09:12:55 PM Engine disposed.
```



Query Query History

```sql
SELECT * FROM public.merged_data
ORDER BY id ASC
```

Data Output Messages Notifications

	id [PK] integer	track_id character varying (255)	artists text	album_name character varying (255)	track_name text	popularity integer	explicit boolean	danceability double precision	energy double precision	track_genr character v
1	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	false	0.676	0.461	Instrumen
2	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	false	0.42	0.166	Instrumen
3	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	false	0.438	0.359	Instrumen
4	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Soundtrack)	Can't Help Falling In Love	71	false	0.266	0.0596	Instrumen
5	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	false	0.618	0.443	Instrumen
6	5	01MVOl9KtVTNfFiBU9I7dc	Tyrone Wells	Days I Will Remember	Days I Will Remember	58	false	0.688	0.481	Instrumen
7	6	6Vc5wAMmXdKIAM7WUoE...	A Great Big World;Christi...	Is There Anybody Out There?	Say Something	74	false	0.407	0.147	Instrumen
8	7	1EzrEOXmMH3G43AXT1y7...	Jason Mraz	We Sing. We Dance. We Steal Things.	I'm Yours	80	false	0.703	0.444	Instrumen
9	8	0IktbUcnAGrvD03AWnz3Q8	Jason Mraz;Colbie Caillat	We Sing. We Dance. We Steal Things.	Lucky	74	false	0.625	0.414	Instrumen
10	9	7k9GuJYLp2AzqokyEdwEw2	Ross Copperman	Hunger	Hunger	56	false	0.442	0.632	Instrumen
11	10	4mzP5mHkRvGxdhdGdAH7...	Zack Tabudlo	Episode	Give Me Your Forever	74	false	0.627	0.363	Instrumen
12	11	5ivF4eQBqJiVL5IAE9jRyl	Jason Mraz	Love Is a Four Letter Word	I Won't Give Up	69	false	0.483	0.303	Instrumen
13	12	4ptDJbJl35d7gQfeNteBwp	Dan Berk	Solo	Solo	52	false	0.489	0.314	Instrumen
14	13	0X9MxHR1rTkEHDjp95F2OO	Anna Hamilton	Bad Liar	Bad Liar	62	false	0.691	0.234	Instrumen
15	14	4LbWtBkN82ZRhz9jqzgrb3	Chord Overstreet;Deepend	Hold On (Remix)	Hold On - Remix	56	false	0.755	0.78	Instrumen
16	15	1KHdq8NK9QxnGjdXb55NiG	Landon Pigg	The Boy Who Never	Falling in Love at a Coffee Shop	58	false	0.489	0.561	Instrumen
17	16	6xKeQgzfjixSUld14qUezm	Andrew Foy;Renee Foy	ily (i love you baby)	ily (i love you baby)	56	false	0.706	0.112	Instrumen
18	17	4Yo0igmcoNyat1secaH0OD	Andrew Foy;Renee Foy	At My Worst	At My Worst	54	false	0.795	0.0841	Instrumen
19	18	6CgNoAbFJ4Q4Id4EjtbXIC	Boyce Avenue;Bea Miller	Cover Sessions, Vol. 4	Photograph	67	false	0.717	0.32	Instrumen
20	19	210JCw2LbYD4YIs8GiZ9iP	Boyce Avenue;Jennel Gar...	Cover Sessions, Vol. 3	Demons	63	false	0.678	0.351	Instrumen
21	20	6D33wCKzWtNEgOovgeVJ7r	Jason Mraz	Mellow Adult Pop	Bella Luna	1	false	0.755	0.454	Instrumen

Total rows: 1000 of 81941 Query complete 00:00:03.238 Ln 1, Col 1

duration_category character varying (255)	popularity_category character varying (255)	track_mood character varying (255)	live_performance boolean	title character varying (255)	category character varying (255)	is_nominated boolean
Average	High Popularity	Happy	false	Not applicable	Not applicable	false
Short	Average Popularity	Sad	false	Not applicable	Not applicable	false
Average	Average Popularity	Sad	false	Not applicable	Not applicable	false
Average	High Popularity	Sad	false	Not applicable	Not applicable	false
Average	High Popularity	Sad	false	Not applicable	Not applicable	false
Average	Average Popularity	Happy	false	Not applicable	Not applicable	false
Average	High Popularity	Sad	false	57th Annual GRAMMY Awards (2014)	Best Pop Duo/Group Performance	true
Average	High Popularity	Happy	false	Not applicable	Not applicable	false
Average	High Popularity	Happy	false	52nd Annual GRAMMY Awards (200...	Best Pop Collaboration With Vocals	true
Average	Average Popularity	Sad	false	Not applicable	Not applicable	false
Average	High Popularity	Happy	false	Not applicable	Not applicable	false
Average	Average Popularity	Sad	false	Not applicable	Not applicable	false
Average	Average Popularity	Happy	false	44th Annual GRAMMY Awards (2001)	Best Rock Gospel Album	true
Average	Average Popularity	Sad	false	Not applicable	Not applicable	false
Average	Average Popularity	Neutral	false	Not applicable	Not applicable	false
Average	Average Popularity	Sad	false	Not applicable	Not applicable	false
Short	Average Popularity	Neutral	false	Not applicable	Not applicable	false
Average	Average Popularity	Happy	false	Not applicable	Not applicable	false
Average	Average Popularity	Neutral	false	Not applicable	Not applicable	false
Average	Average Popularity	Neutral	false	Not applicable	Not applicable	false
Long	Low Popularity	Neutral	false	Not applicable	Not applicable	false

Storing the data

It's important to note the configuration files and credentials required by this module before starting the Airflow process:

- *credentials/client_secrets.json*
- *credentials/saved_credentials.json*
- *env/settings.yaml*

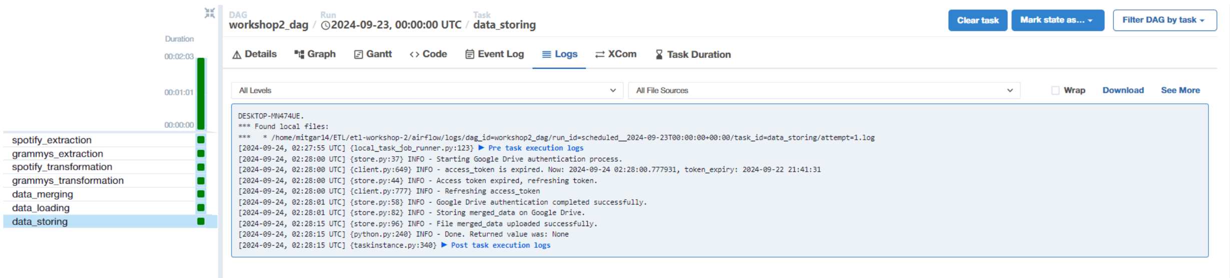
In addition, you must specify in the .env file the locations of these files with their respective absolute paths and the Google Drive ID of the folder in which you want to insert the file.

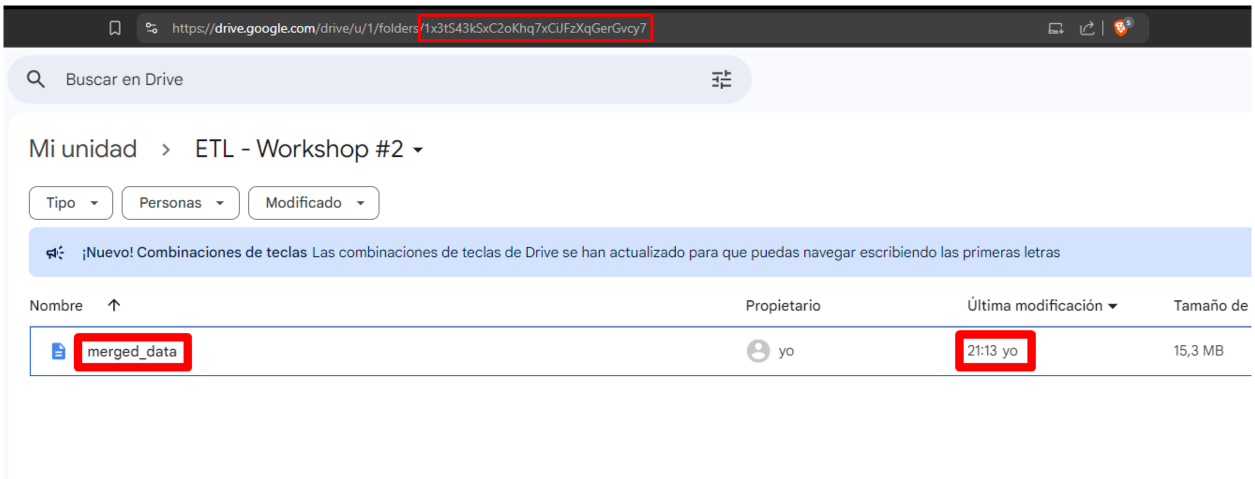
```
# Google Drive variables

CLIENT_SECRETS_PATH = "/home/mitgar14/ETL/etl-workshop-2/credentials/client_secrets.json"
SETTINGS_PATH = "/home/mitgar14/ETL/etl-workshop-2/env/settings.yaml"
SAVED_CREDENTIALS_PATH = "/home/mitgar14/ETL/etl-workshop-2/credentials/saved_credentials.json"

FOLDER_ID = 1x3tS43kSxC2oKhq7xCiJFzXqGerGvcy7
```

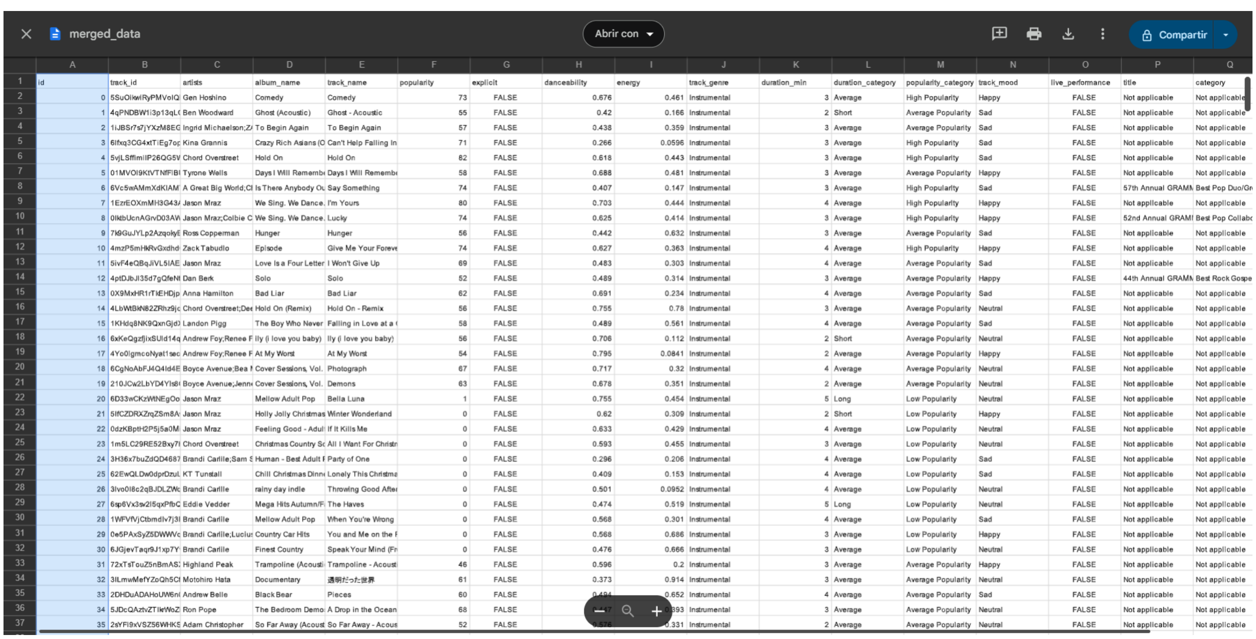
In [19]:

```
storing_merged_data("merged_data", merged_df)
```

```
23/09/2024 09:12:55 PM Starting Google Drive authentication process.
23/09/2024 09:12:55 PM access_token is expired. Now: 2024-09-24 02:12:55.993523, token_expiry: 2024-09-20 20:52:25
23/09/2024 09:12:55 PM Access token expired, refreshing token.
23/09/2024 09:12:56 PM Refreshing access_token
23/09/2024 09:12:56 PM Google Drive authentication completed successfully.
23/09/2024 09:12:56 PM Storing merged_data on Google Drive.
23/09/2024 09:13:10 PM File merged_data uploaded successfully.
```

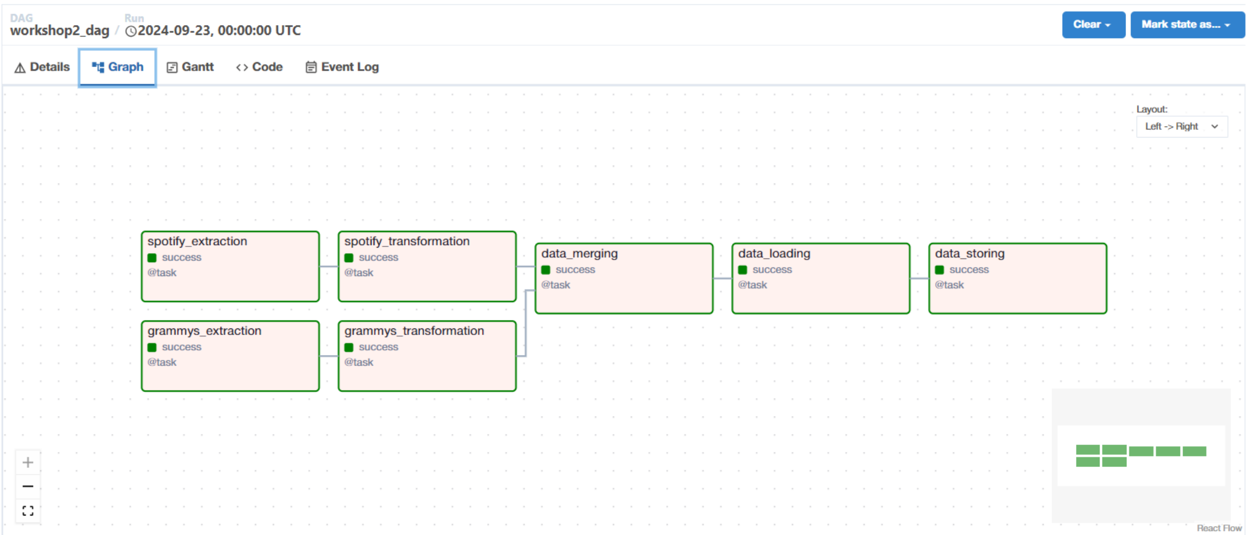






Managing the data pipeline with Airflow

In this workshop it's used the DAG workshop2_dag. The DAG code was realized using the Taskflow API, therefore, the tasks and context statement of our DAG is composed of decorators.



Analyzing the composition of the DAG

The image in this section belongs to the DAG declaration of this workshop. As can be seen, we use decorators to declare that the following functions are our tasks.

```python
def workshop2_dag():

    @task
    def spotify_transformation(raw_df):
        return transform_spotify(raw_df)

    spotify_data = spotify_transformation(spotify_raw_data)

    @task
    def grammys_transformation(raw_df):
        return transform_grammys(raw_df)

    grammys_data = grammys_transformation(grammys_raw_data)

    @task
    def data_merging(spotify_df, grammys_df):
        return merge_data(spotify_df, grammys_df)

    df = data_merging(spotify_data, grammys_data)

    @task
    def data_loading(df):
        return load_data(df)

    data_load = data_loading(df)

    @task
    def data_storing(df):
        return store_data(df)

    data_store = data_storing(data_load)

workshop2_dag = workshop2_dag()
```

```python
# Creating tasks functions
# -----------------------

def extract_spotify():
    try:
        df = extracting_spotify_data("./data/spotify_dataset.csv")
        return df.to_json(orient="records")
    except Exception as e:
        logging.error(f"Error extracting data: {e}")

def extract_grammys():
    try:
        df = extracting_grammys_data()
        return df.to_json(orient="records")
    except Exception as e:
        logging.error(f"Error extracting data: {e}")

def transform_spotify(df):
    try:
        json_df = json.loads(df)

        raw_df = pd.DataFrame(json_df)
        df = transforming_spotify_data(raw_df)

        return df.to_json(orient="records")
    except Exception as e:
        logging.error(f"Error transforming data: {e}")

def transform_grammys(df):
    try:
        json_df = json.loads(df)

        raw_df = pd.DataFrame(json_df)
        df = transforming_grammys_data(raw_df)

        return df.to_json(orient="records")
    except Exception as e:
        logging.error(f"Error transforming data: {e}")
```

However, our DAG functions call functions that are not even part of that file.

The functions that are called in that file belong to the `etl` module, which is part of *tasks*.

One important thing to note is that in order to receive and send data between tasks, the functions will receive and convert the data to JSON, one of the few data formats allowed in Airflow.

Another thing to note is that the call chain does not stop here: these functions call more functions. These other functions are part of the various packages and modules that make up `src`.

Visualizing the Data

The Power BI dashboard provides a comprehensive visual representation of data related to songs, genres, and artists on Spotify. It enables users to quickly identify patterns and gain deeper insights into the music trends on Spotify.

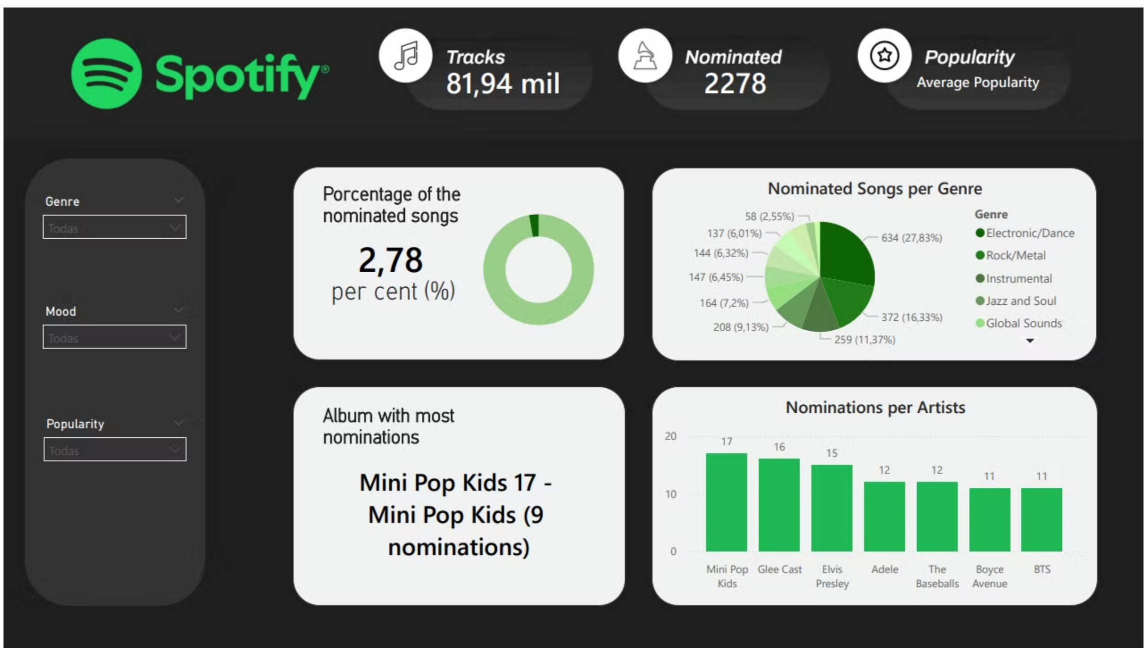
Below is a detailed breakdown of each page of the dashboard and its key elements.

Page 1: Nominations

This first page of the dashboard gives an overview of song popularity and the number of nominations received by artists and albums. Key visualizations include:

- **Average Popularity**: A KPI visual showing the average popularity score across all songs in the dataset (e.g., 81.94). This provides a quick snapshot of how well songs are performing overall.

- **Nominations per Artist**: A bar chart that displays the number of nominations received by different artists. For example, artists like BTS, Adele, and Elvis Presley have notable mentions, allowing users to see which artists are leading in nominations.
- **Nominated Songs per Genre**: A bar chart that shows the number of nominated songs in each genre. Genres like Electronic/Dance, Rock/Metal, and Jazz and Soul are prominently featured, giving insight into which genres dominate nominations.
- **Album with Most Nominations**: A card visual showing the album with the highest number of nominations, such as *Mini Pop Kids 17*, which has 9 nominations, making it the top-nominated album in this dataset.

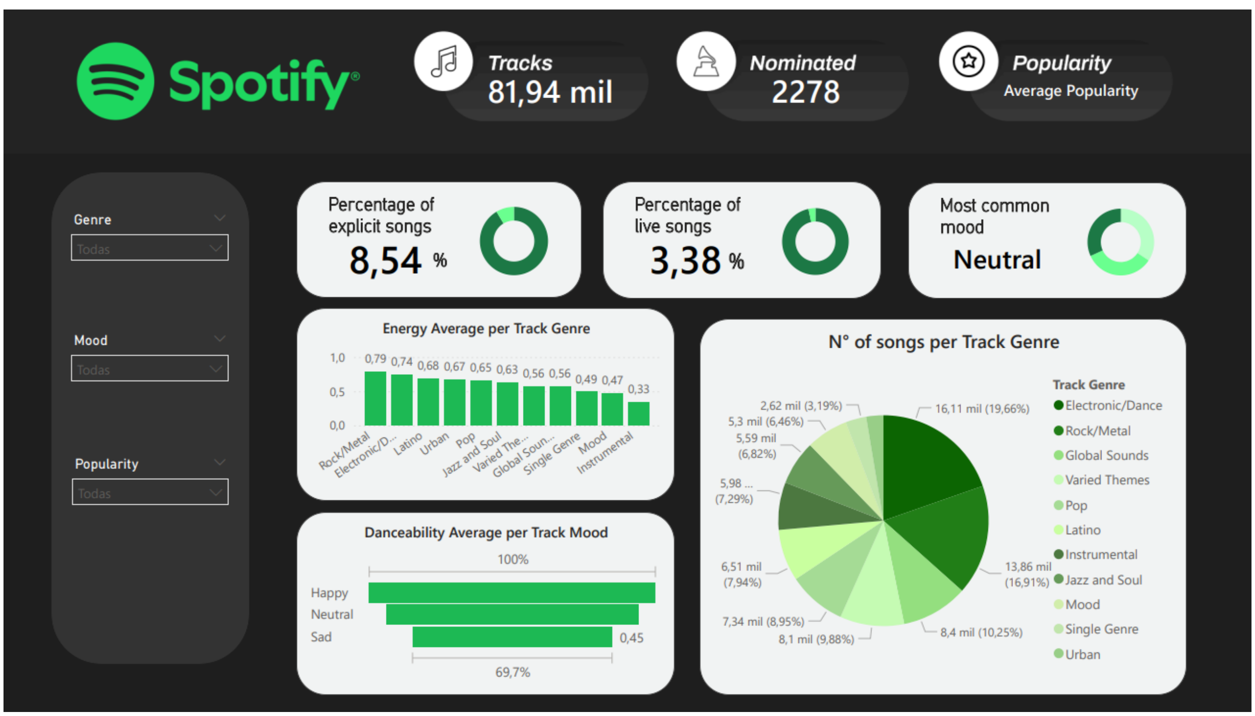


Page 2: Song Characteristics

The second page dives deeper into the specific characteristics of songs, such as energy levels, explicit content, and live performances. Key metrics and visualizations include:

- **Percentage of Explicit Songs**: A KPI that displays the percentage of songs marked as explicit, providing a clear view of how much explicit content is present in the dataset (e.g., 8.54%).

- **Percentage of Live Songs**: Another KPI showing the percentage of live recordings within the dataset (e.g., 3.38%). This helps differentiate between studio recordings and live performances.
- **Most Common Mood**: A card visual showing the most common mood found in the tracks. For instance, "Neutral" may be highlighted as the predominant mood, giving an idea of the emotional tone of the majority of songs.
- **Energy Average per Track Genre**: A bar chart that compares the average energy level of tracks across different genres. Genres like Rock/Metal, Electronic/Dance, and Jazz and Soul are shown with varying energy averages, giving insights into the intensity of songs in each category.
- **Danceability Average per Track Mood**: A bar chart that displays the average danceability score based on the mood of the song. Moods such as Happy, Neutral, and Sad are analyzed, allowing users to understand how different emotional tones impact the danceability of tracks.
- **Number of Songs per Track Genre**: A chart that shows the total number of songs within each genre, providing an overview of the distribution of genres in the dataset. Genres like Pop, Electronic/Dance, and Rock/Metal may have the highest representation.



Conclusions

1. **Successful Data Integration:** The workshop demonstrated effective merging of Spotify and Grammy datasets, highlighting the importance of data cleaning and transformation in preparing datasets for analysis.
2. **Airflow Pipeline Efficiency:** The implementation of Apache Airflow showcased the power of automated, scalable data pipelines, essential for handling complex ETL processes in real-world scenarios.
3. **Insightful Visualizations:** The Power BI dashboard provided valuable insights into music trends, artist popularity, and genre distribution, emphasizing the importance of data visualization in extracting meaningful information.
4. **Cross-Platform Data Management:** The workshop illustrated the seamless integration of various tools (Python, PostgreSQL, Airflow, Power BI) in a comprehensive data analysis workflow, reflecting real-world data engineering practices.

6. **Industry Relevance:** By analyzing music industry data, the workshop provided practical experience in handling and interpreting real-world datasets, preparing participants for similar challenges in professional settings.