WORKSHOP 2

MARÍA FERNANDA TELLO VERGARA

2225338

JAVIER ALEJANDRO VERGARA

ETL

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

OCTUBRE 04 2024

CONTEXT

We are going to extract information using differents data sources (csv file, database), then make some transformations and merge the transformed data to finally upload it to google drive as a CSV file and store the data in a DB. As a last step, create a dashboard from the data stored in the DB to visualize the information in the best way you consider.

DESCRIPTION

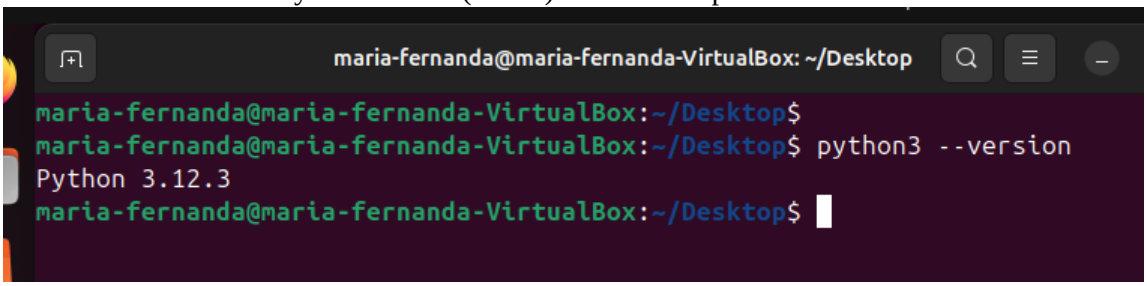
We will use the Spotify dataset to read it in Python and Airflow, create some transformations and load it into a database, on the other hand we will use the Grammys dataset to load it into a database, then using Airflow we will read the data from the database, perform some transformations, merge with the Spotify dataset and load into the database.

TOOLS

• Python (Pandas, Matplotlib, Sqlalchemy dotenv).
• Postgress.
• Jupyter Notebook.
• Datasets.
• Docker
• Encryption of credentials using a .env file (Environment variables).
• Apache airflow.
• Google drive.
• Power BI.

STEP BY STEP

We check the installed Python version (3.12.3) and install Apache Airflow version 2.10.1.




```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$ python3 --version
Python 3.12.3
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$ pip install "apache-airflow[
celery]==2.10.1" --constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-2.10.1/constraints-3.8.txt"
```

We create the workshop2 directory, initialize a Git repository and configure a virtual environment in Python.

```
-system-packages.
hint: See PEP 668 for the detailed specification.
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$ mkdir workshop2
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$ ls
workshop2
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$ cd workshop2
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ ls
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ git init
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ rm -rf venv
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ python3 -m venv venv
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ ls
venv
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$
```

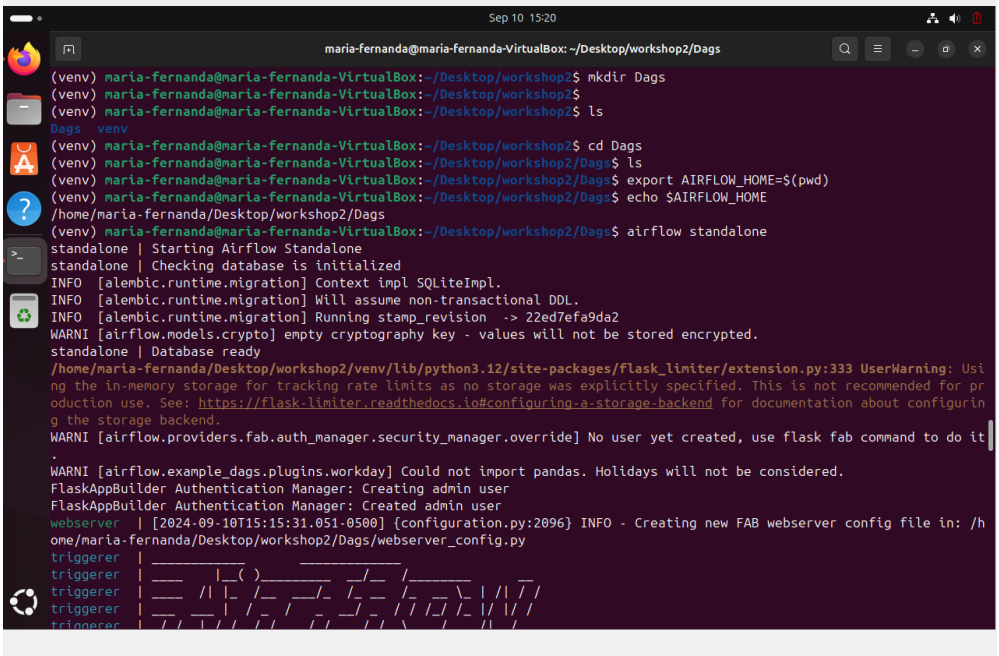
We enter the virtual environment:

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ source venv/bin/activate
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$
```

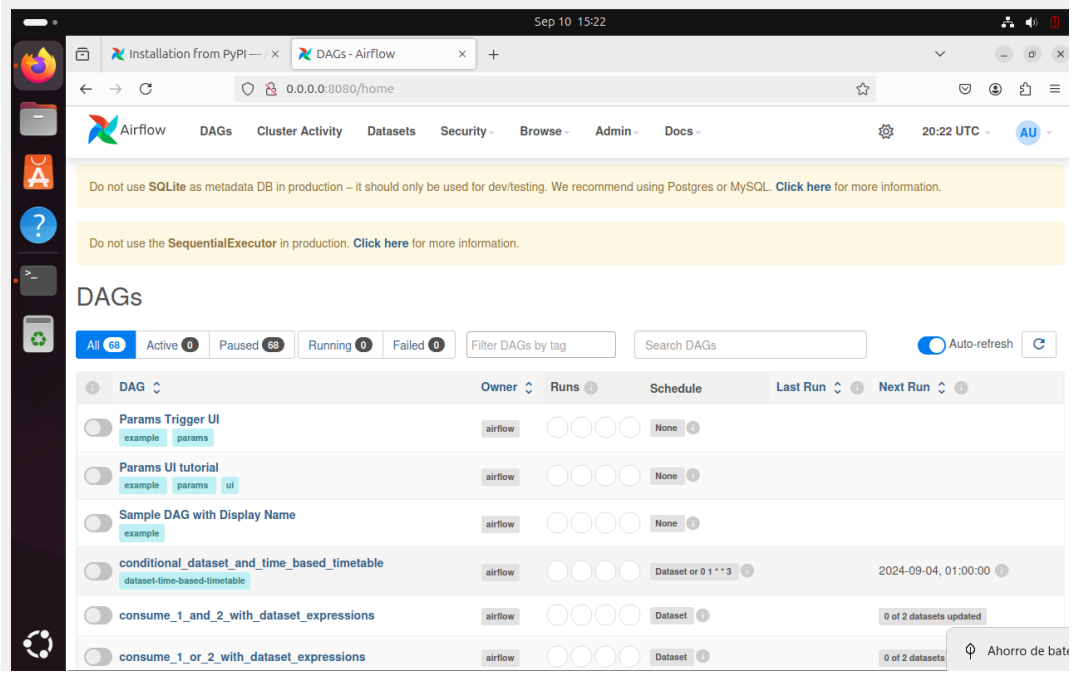
We created the Dags directory, logged into it, and set the AIRFLOW_HOME environment variable to the current path of the Dags directory.

```
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ mkdir Dags
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ ls
Dags  venv
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ cd Dags
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ ls
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ export AIRFLOW_HOME=$(pwd)
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ echo $AIRFLOW_HOME
/home/maria-fernanda/Desktop/workshop2/Dags
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
```

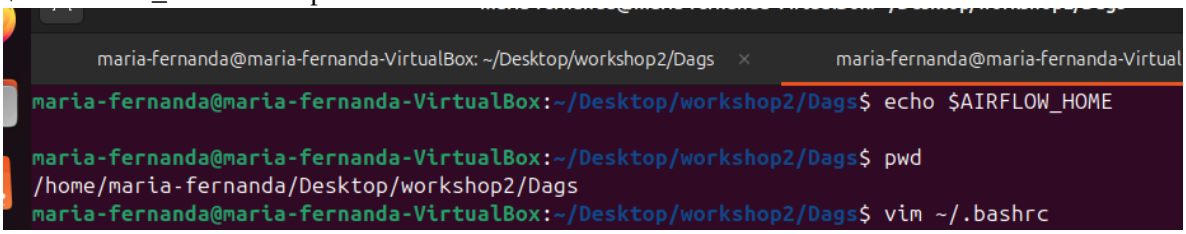
We start Airflow in standalone mode after setting the AIRFLOW_HOME environment variable. The process includes initializing the database, creating an admin user, and configuring the Airflow web server.

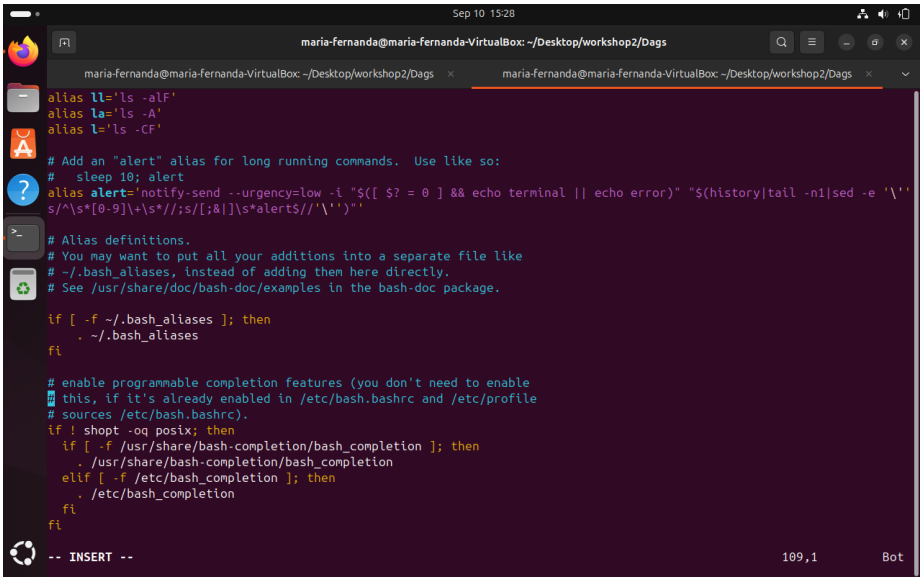


We open the Airflow web interface in the browser, showing the list of available DAGs (workflows).



We check the AIRFLOW_HOME environment variable and its path with the echo $AIRFLOW_HOME and pwd commands.





Set the variable in the system to not be cleared.

```
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export AIRFLOW_HOME=/home/maria-fernanda/Desktop/workshop2/Dags
```

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ vim ~/.bashrc
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name 'MafeTello'
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name
MafeTello
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
```

Edited Git configurations to set the global username to MafeTello using the git config user.name command.

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ vim ~/.bashrc
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name 'MafeTello'
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name
MafeTello
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.email
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.email 'maria_fernanda.tello@uao.edu.c
o'
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.email
maria_fernanda.tello@uao.edu.co
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
```

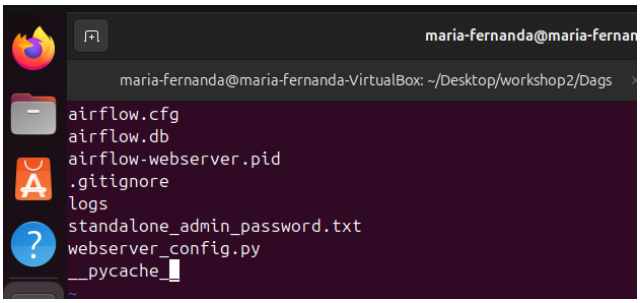
We configured the Git email with git config user.email, setting maria_fernanda.tello@uao.edu.co. Then, we added the remote repository to Git with git remote add origin, pointing to the repository on GitHub: https://github.com/MafeTello/workshop2.git. Finally, the connection to the remote repository was verified with git remote -v and a .gitignore file was created.

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ vim ~/.bashrc
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name 'MafeTello'
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.name
MafeTello
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.email
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.email 'maria_fernanda.tello@uao.edu.c
o'
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git config user.email
maria_fernanda.tello@uao.edu.co
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git remote add origin https://github.com/MafeTello/wo
rkshop2.git
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git remote -v
origin  https://github.com/MafeTello/workshop2.git (fetch)
origin  https://github.com/MafeTello/workshop2.git (push)
```

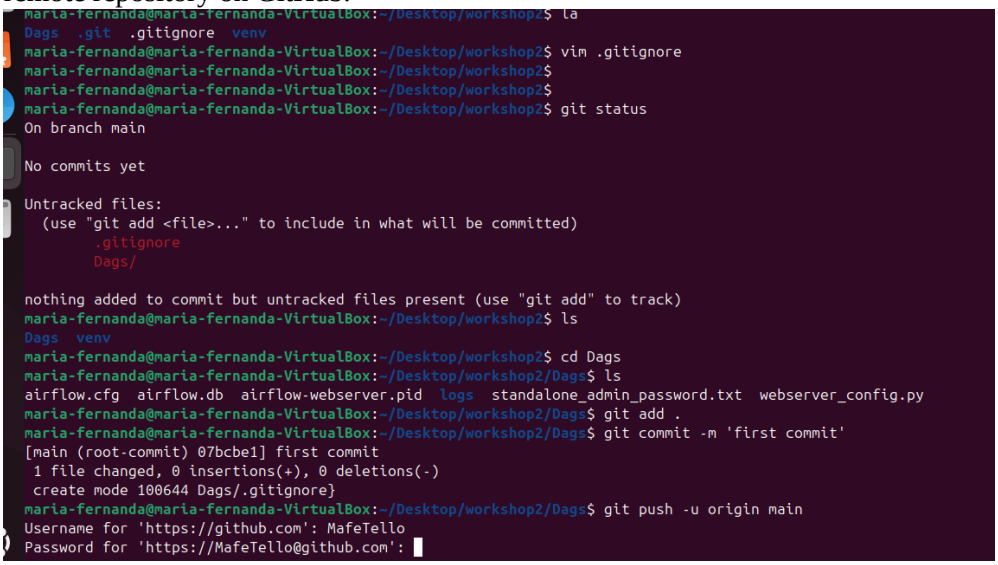
```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ vim .gitignore
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
```

(We did not upload these files to the repository)

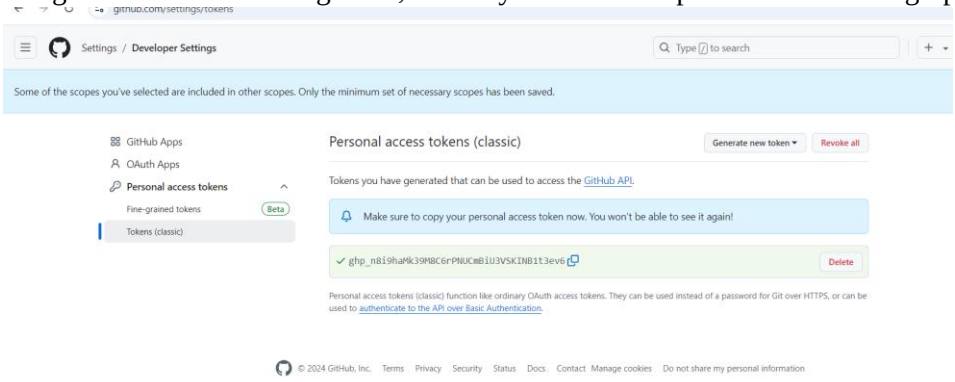
We added the pycache.



Here we made the Git configurations, made the first commit and uploaded the changes to the remote repository on GitHub.



We generate the tokens in git hub, and they are used as a password to do the git push.



The changes were pushed to the remote repository on GitHub, creating and linking the main branch with origin/main.

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$ git push -u origin main
Username for 'https://github.com': MafeTello
Password for 'https://MafeTello@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Writing objects: 100% (4/4), 261 bytes | 261.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MafeTello/workshop2.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/Dags$
```

Install postgreSQL on Ubuntu.

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ sudo apt install postgresql postgresql-contrib
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libpq5 libtypes-serialiser-perl postgresql-16 postgresql-client-16
  postgresql-client-common postgresql-common
Suggested packages:
  postgresql-doc postgresql-doc-16
The following NEW packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libpq5 libtypes-serialiser-perl postgresql postgresql-16
  postgresql-client-16 postgresql-client-common postgresql-common postgresql-contrib
0 upgraded, 11 newly installed, 0 to remove and 8 not upgraded.
Need to get 17.3 MB of archives.
After this operation, 50.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://co.archive.ubuntu.com/ubuntu noble/main amd64 libjson-perl all 4.10000-1 [81.9 kB]
Get:2 http://co.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-client-common all 257build1.1 [36.4 kB]
Get:3 http://co.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-common all 257build1.1 [161 kB]
Get:4 http://co.archive.ubuntu.com/ubuntu noble/main amd64 libcommon-sense-perl amd64 3.75-3build3 [20.4 kB]
Get:5 http://co.archive.ubuntu.com/ubuntu noble/main amd64 libtypes-serialiser-perl all 1.01-1 [11.6 kB]
```

We accessed the PostgreSQL console with the command sudo -i -u postgres.

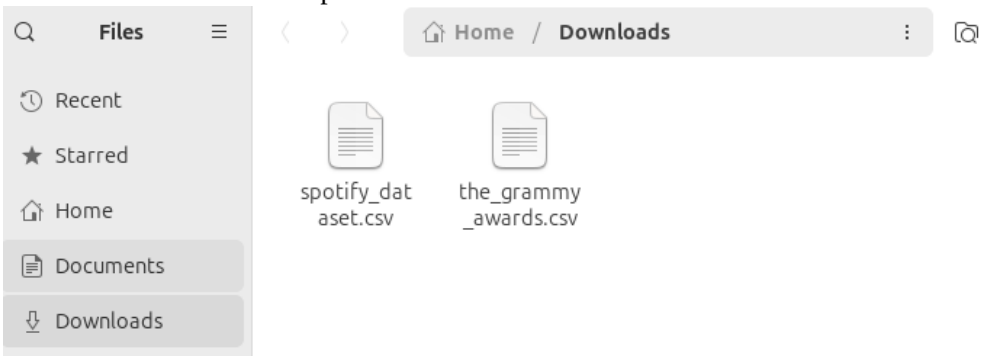
```
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ sudo systemctl start postgresql.service
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled; preset: enabled)
     Active: active (exited) since Tue 2024-09-10 16:21:22 -05; 50s ago
   Main PID: 18424 (code=exited, status=0/SUCCESS)
        CPU: 2ms

Sep 10 16:21:22 maria-fernanda-VirtualBox systemd[1]: Starting postgresql.service - PostgreSQL RDBMS...
Sep 10 16:21:22 maria-fernanda-VirtualBox systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ sudo -i -u postgres
postgres@maria-fernanda-VirtualBox:~$ psql
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
Type "help" for help.

postgres=#
```

\q (salir)

PostgreSQL port settings and machine IP address are verified

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ sudo -i -u postgres
postgres@maria-fernanda-VirtualBox:~$ psql
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
Type "help" for help.

postgres=# exit
postgres@maria-fernanda-VirtualBox:~$ psql
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
Type "help" for help.

postgres=# show port;
 port
------
 5432
(1 row)

postgres=# exit
postgres@maria-fernanda-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe5f:f5eb  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:5f:f5:eb  txqueuelen 1000  (Ethernet)
        RX packets 189138  bytes 261617114 (261.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23875  bytes 4405403 (4.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

We download the files to upload them to the database.



A table named grammy_awards was created in PostgreSQL with several columns. The database relations were then listed using the \d command, confirming the existence of the table, and the detailed structure of the table was displayed using \d grammy_awards.

```
postgres=# CREATE TABLE grammy_awards (
    year INT,
    title VARCHAR(255),
    published_at TIMESTAMP,
    updated_at TIMESTAMP,
    category VARCHAR(255),
    nominee VARCHAR(255),
    artist VARCHAR(255),
    link TEXT,
    image TEXT,
    winner BOOLEAN
);
CREATE TABLE
postgres=#
```

```
postgres=# \d
              List of relations
 Schema |     Name      | Type  |  Owner
--------+---------------+-------+----------
 public | grammy_awards | table | postgres
(1 row)
```

```
postgres=#
postgres=# \d grammy_awards
                        Table "public.grammy_awards"
    Column    |            Type             | Collation | Nullable | Default
--------------+-----------------------------+-----------+----------+---------
 year         | integer                     |           |          |
 title        | character varying(255)      |           |          |
 published_at | timestamp without time zone |           |          |
 updated_at   | timestamp without time zone |           |          |
 category     | character varying(255)      |           |          |
 nominee      | character varying(255)      |           |          |
 artist       | character varying(255)      |           |          |
 link         | text                        |           |          |
 image        | text                        |           |          |
 winner       | boolean                     |           |          |

postgres=#
```

We give read and write permissions so that any user can read and write the file.

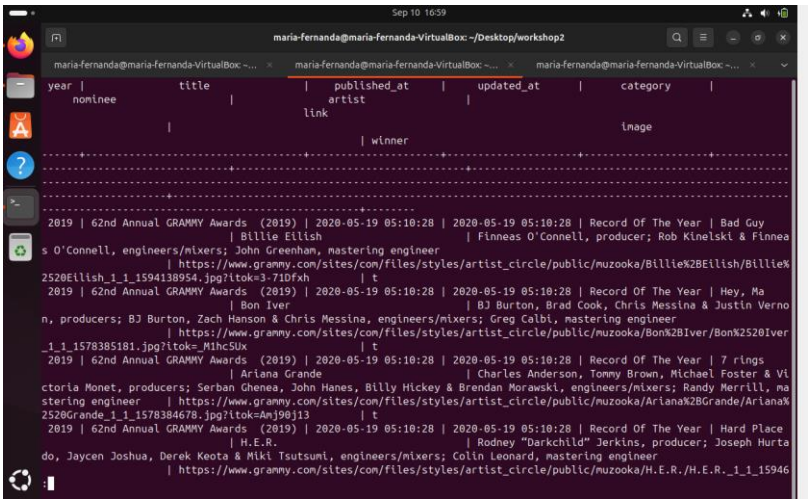
```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ ls
Dags  venv
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ cd ..
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$ ls
workshop2
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop$ cd ..
maria-fernanda@maria-fernanda-VirtualBox:~$ ls
airflow  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
maria-fernanda@maria-fernanda-VirtualBox:~$ pwd
/home/maria-fernanda
maria-fernanda@maria-fernanda-VirtualBox:~$ cd Downloads/
maria-fernanda@maria-fernanda-VirtualBox:~/Downloads$ ls
spotify_dataset.csv  the_grammy_awards.csv
maria-fernanda@maria-fernanda-VirtualBox:~/Downloads$ ll
total 21092
drwxr-xr-x  2 maria-fernanda maria-fernanda     4096 Sep 10 16:30 ./
drwxr-x--- 16 maria-fernanda maria-fernanda     4096 Sep 10 16:00 ../
-rw-rw-r--  1 maria-fernanda maria-fernanda 20118244 Sep 10 16:30 spotify_dataset.csv
-rw-rw-r--  1 maria-fernanda maria-fernanda  1468421 Sep 10 16:29 the_grammy_awards.csv
maria-fernanda@maria-fernanda-VirtualBox:~/Downloads$ chmod 777 the_grammy_awards.csv
maria-fernanda@maria-fernanda-VirtualBox:~/Downloads$ ll
total 21092
drwxr-xr-x  2 maria-fernanda maria-fernanda     4096 Sep 10 16:30 ./
drwxr-x--- 16 maria-fernanda maria-fernanda     4096 Sep 10 16:00 ../
-rw-rw-r--  1 maria-fernanda maria-fernanda 20118244 Sep 10 16:30 spotify_dataset.csv
-rwxrwxrwx  1 maria-fernanda maria-fernanda  1468421 Sep 10 16:29 the_grammy_awards.csv*
maria-fernanda@maria-fernanda-VirtualBox:~/Downloads$
```

We give permissions to home.

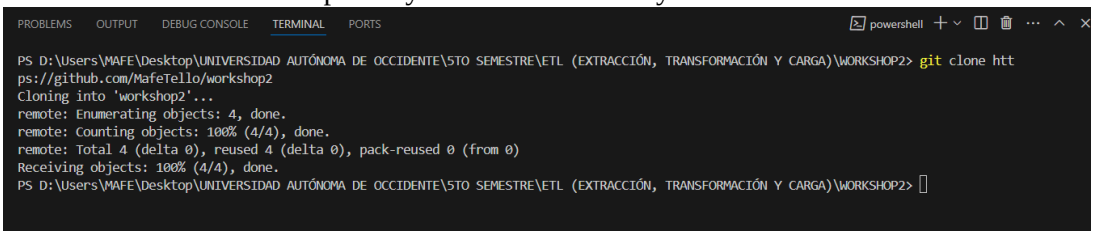
```
-rw-rw-r--  1 maria-fernanda maria-fernanda 20118244 Sep 10 16:30 spotify_dataset.csv
-rw-r--r--  1 maria-fernanda maria-fernanda  1468421 Sep 10 16:29 the_grammy_awards.csv
maria-fernanda@maria-fernanda-VirtualBox:~/Downloads$ sudo chmod 777 /home/maria-fernanda
[sudo] password for maria-fernanda:
maria-fernanda@maria-fernanda-VirtualBox:~/Downloads$
```

```
postgres=# \COPY grammy_awards(year, title, published_at, updated_at, category, nominee, artist, link, image, winner)
FROM '/home/maria-fernanda/Downloads/the_grammy_awards.csv'
DELIMITER ','
CSV HEADER;
COPY 4810
postgres=#
```

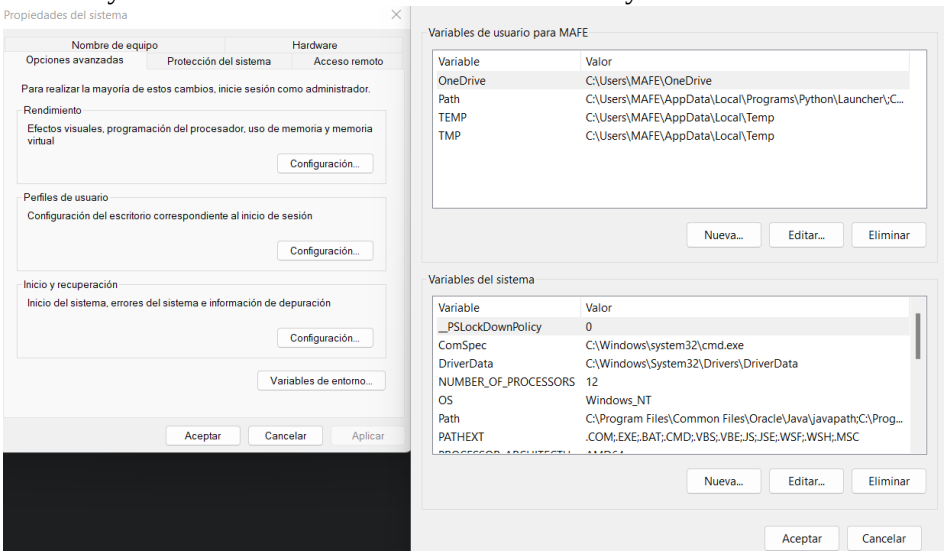
```
postgres=# select * from grammy_awards limit 10
postgres-# ;
postgres=#
```

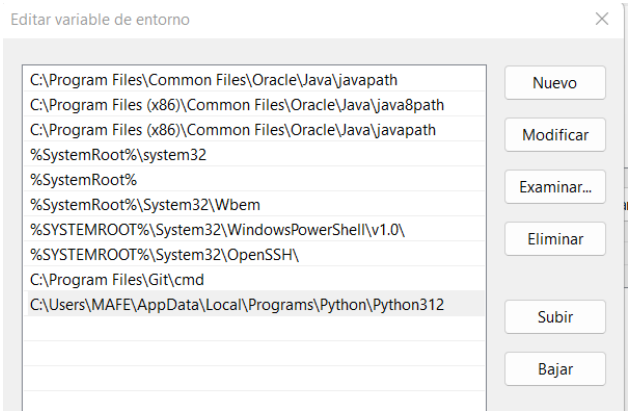


We use git clone to clone the workshop2 repository from GitHub to the local machine, download all the files from the remote repository to the local directory.

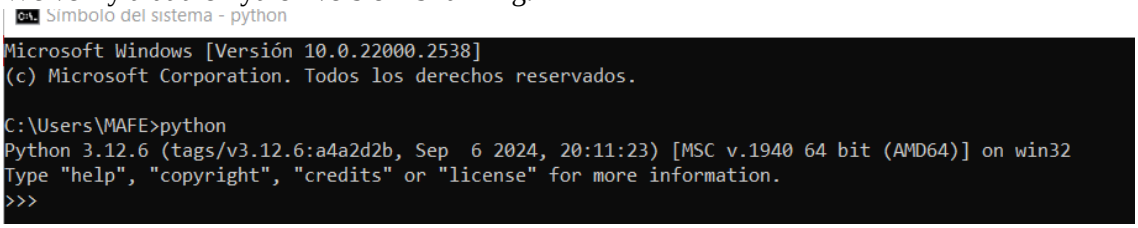


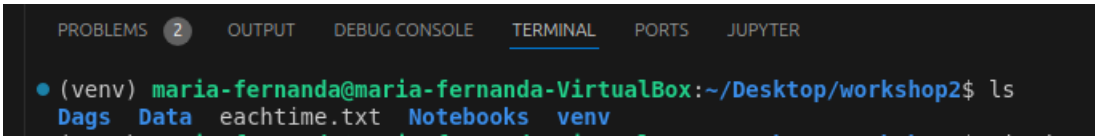
We verify that the environment variables are correctly added.

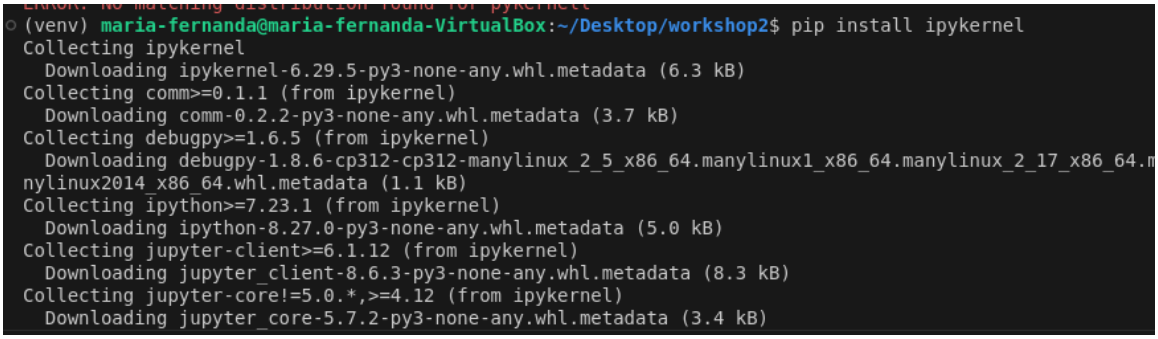


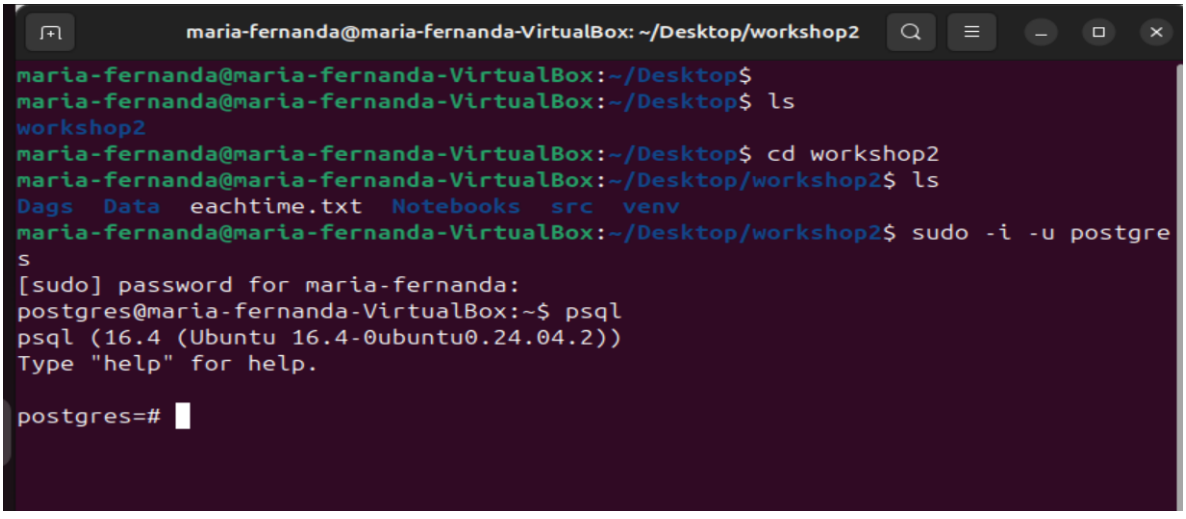


We verify that the Python version is running.

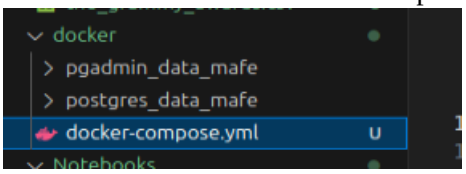








Due to some issues with Windows permissions, I opted to download Docker.



Here we can see that it is already installed.

```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:91fb4b041da273d5a3273b6d587d62d518300a6ad268b28628f74997b93171b2
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash
```

```
db_mafe_test  | 2024-10-01 02:06:47.769 UTC [46] LOG:  background worker "logical replication launcher" (PID 52) exited
with exit code 1
db_mafe_test  | 2024-10-01 02:06:47.770 UTC [47] LOG:  shutting down
db_mafe_test  | 2024-10-01 02:06:47.774 UTC [47] LOG:  checkpoint starting: shutdown immediate
db_mafe_test  | 2024-10-01 02:06:48.097 UTC [47] LOG:  checkpoint complete: wrote 922 buffers (5.6%); 0 WAL file(s) adde
d, 0 removed, 0 recycled; write=0.132 s, sync=0.174 s, total=0.327 s; sync files=301, longest=0.004 s, average=0.001 s;
distance=4255 kB, estimate=4255 kB; lsn=0/1912140, redo lsn=0/1912140
db_mafe_test  | 2024-10-01 02:06:48.104 UTC [46] LOG:  database system is shut down
db_mafe_test  |  done
db_mafe_test  | server stopped
db_mafe_test  |
db_mafe_test  | PostgreSQL init process complete; ready for start up.
db_mafe_test  |
db_mafe_test  | 2024-10-01 02:06:48.213 UTC [1] LOG:  starting PostgreSQL 16.4 (Debian 16.4-1.pgdg120+2) on x86_64-pc-li
nux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db_mafe_test  | 2024-10-01 02:06:48.213 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db_mafe_test  | 2024-10-01 02:06:48.214 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db_mafe_test  | 2024-10-01 02:06:48.221 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_mafe_test  | 2024-10-01 02:06:48.237 UTC [62] LOG:  database system was shut down at 2024-10-01 02:06:48 UTC
db_mafe_test  | 2024-10-01 02:06:48.258 UTC [1] LOG:  database system is ready to accept connections
^CGracefully stopping... (press Ctrl+C again to force)
[+] Stopping 1/1
 ✔ Container db_mafe_test  Stopped                                                                          0.4s
canceled
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/docker$
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/docker$
```

I installed the python-decouple and python-dotenv packages, which are commonly used to handle environment variables in Python projects.

```
    /home/maria-fernanda/Desktop/workshop2/venv/lib/python3.12/site-packages/decouple/*
Proceed (Y/n)? y
  Successfully uninstalled decouple-0.0.7
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ pip install python-decouple python-dotenv
Collecting python-decouple
  Downloading python_decouple-3.8-py3-none-any.whl.metadata (14 kB)
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Downloading python_decouple-3.8-py3-none-any.whl (9.9 kB)
Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-decouple, python-dotenv
Successfully installed python-decouple-3.8 python-dotenv-1.0.1
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$
```

Here we can see that the container is already running

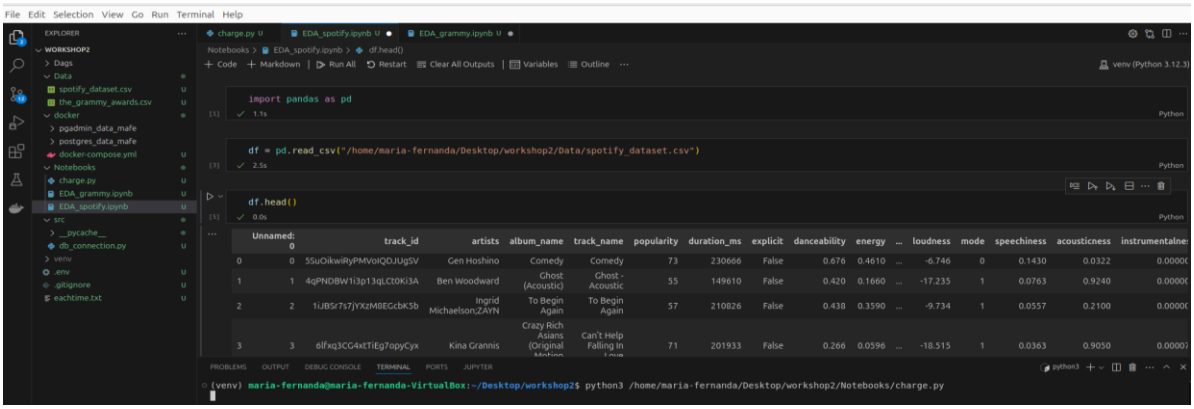
```
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/docker$ sudo docker
 compose up -d
[sudo] password for maria-fernanda:
[+] Running 1/1
 ✔ Container db_mafe_test  Started                                    0.8s
maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/docker$
```

We execute the project.

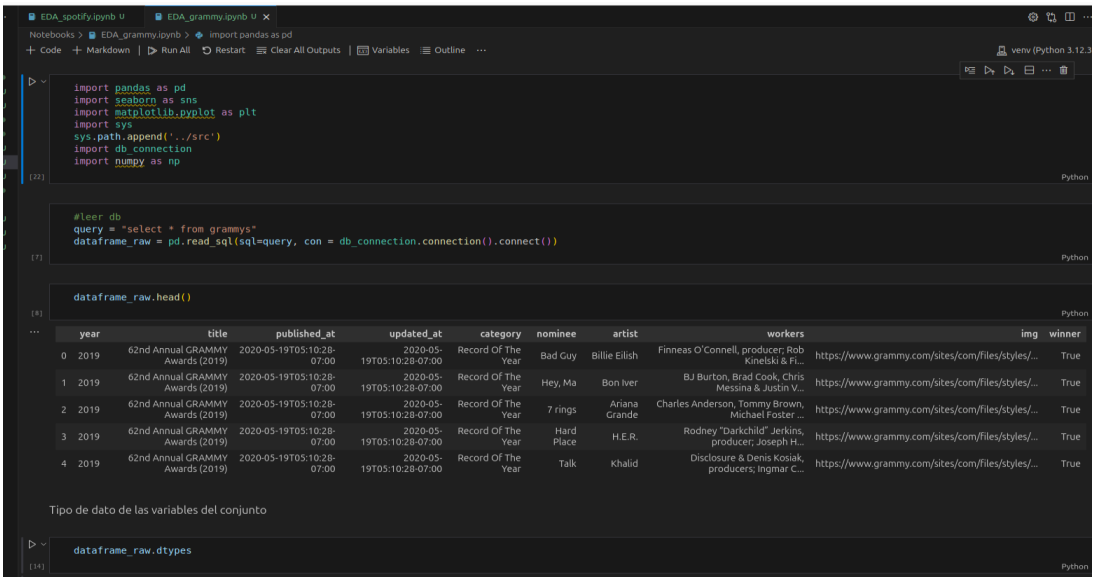
```
  Successfully installed sqlalchemy-2.0.35
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ python3 /home/maria-fernanda/Desktop/workshop2/Notebooks/charge.py
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$
```

Following this we identify the transformations that we must make in the EDA of both datasets.

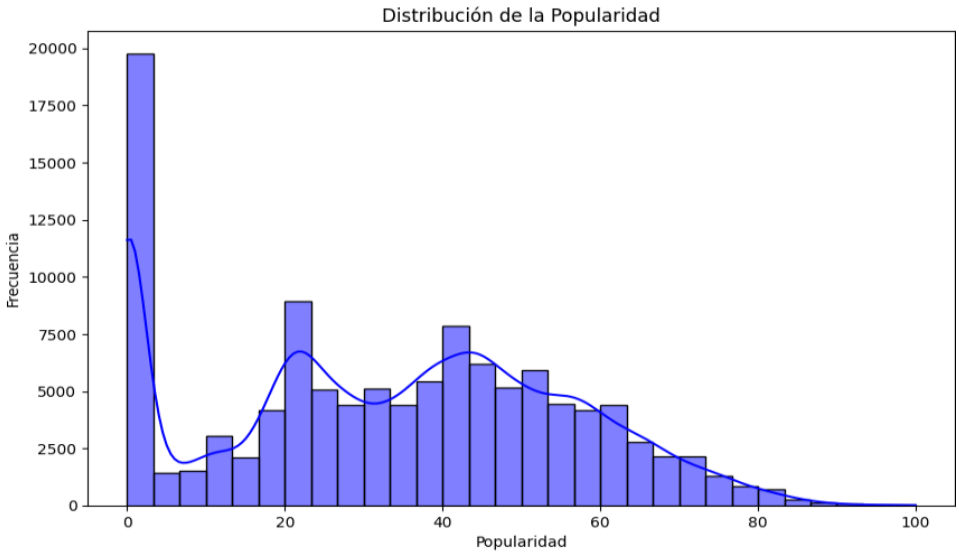
EDA_grammy



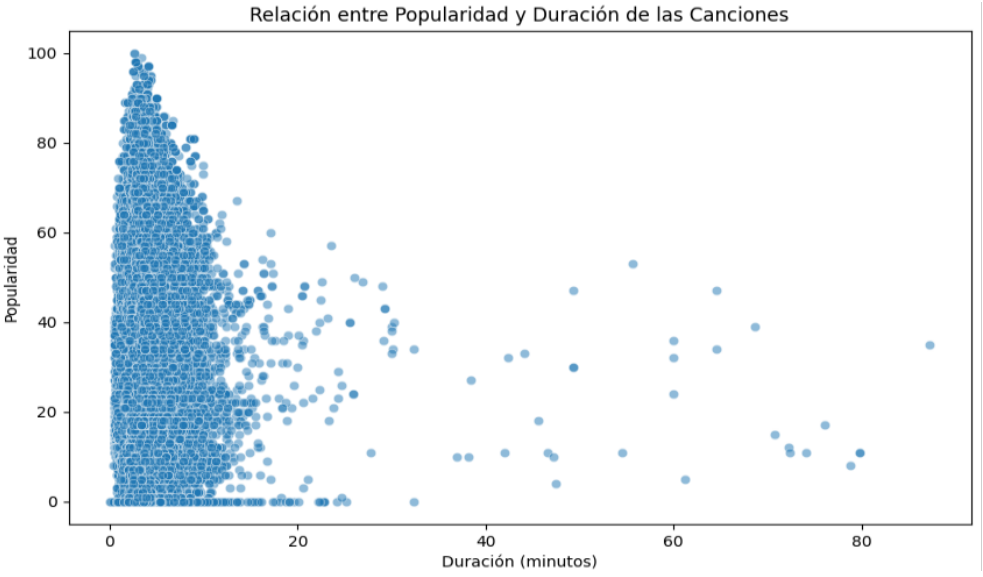
EDA_spotify



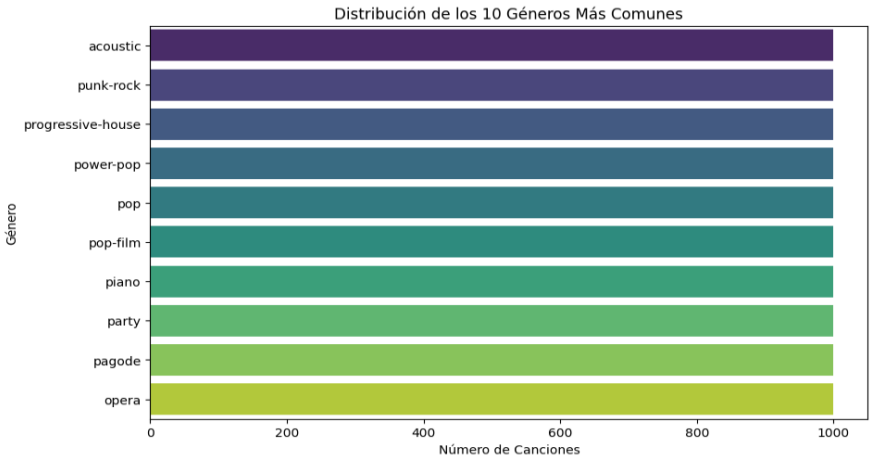
Eda visualizations of spotify dataset



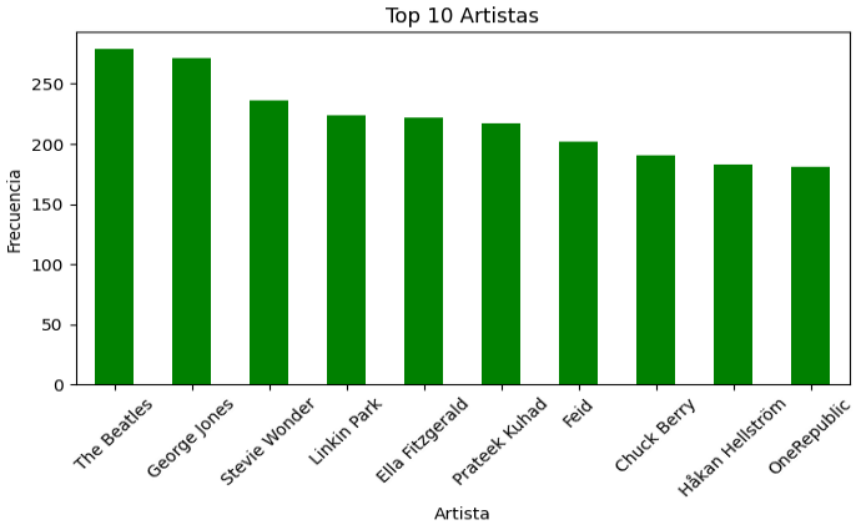
This graph shows the popularity distribution of songs in the dataset. Most songs have low popularity (around 0-20), with a significant peak in the 0-10 range, indicating that a large number of songs are unpopular. However, a more dispersed distribution is observed towards higher values, showing that some songs reach moderate and high popularity (up to 100), although in smaller numbers.



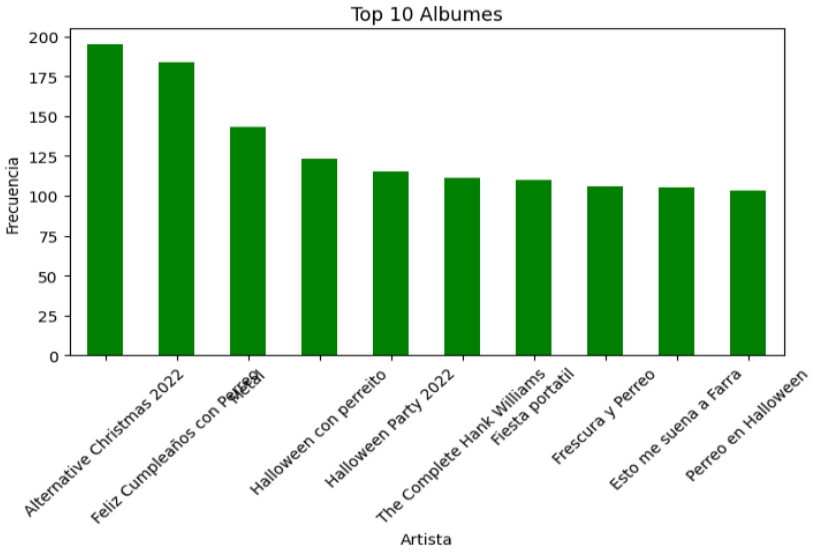
This scatter plot shows the relationship between popularity and song length. Most songs are between 2 and 5 minutes long, regardless of their popularity. There doesn't seem to be a clear correlation between length and popularity, as songs with different lengths have popularity ranging from 0 to 100. However, songs with extremely long lengths (more than 10 minutes) tend to have lower popularity.



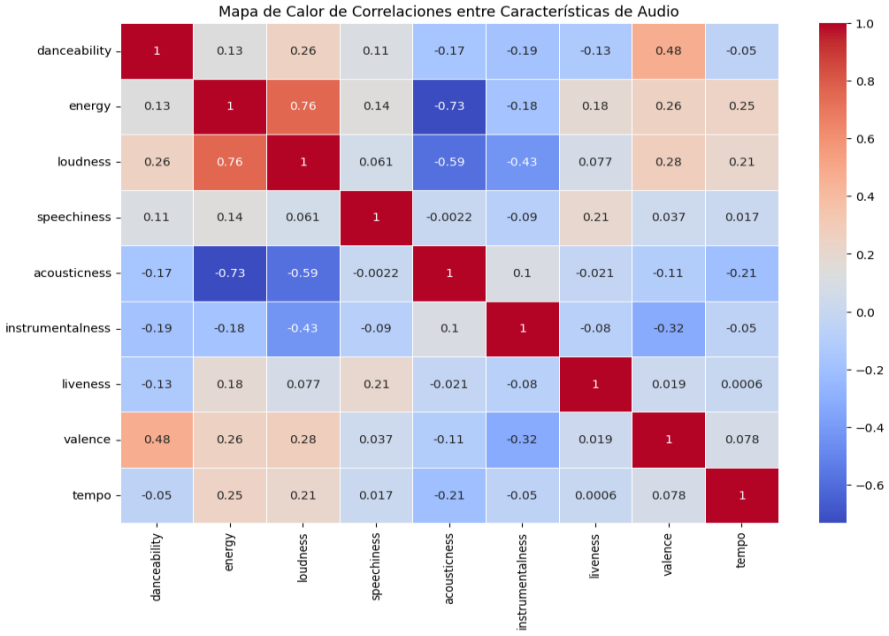
This bar chart shows the distribution of the 10 most common music genres in the Spotify dataset. It can be seen that the genres "acoustic" and "punk-rock" are the most represented, with over 1000 songs each. As we go down the list, genres like "opera" and "pagode" are present, but with a smaller number of songs.



The bar chart shows the top 10 most frequent artists in the Spotify dataset, with The Beatles and George Jones standing out as the most represented, with around 250 songs each. They are followed by Stevie Wonder, Linkin Park, and Ella Fitzgerald, with over 200 songs. Other artists such as Prateek Kuhad, Feid, and Chuck Berry also have a notable presence in the dataset, although in smaller numbers.



The bar chart shows the top 10 albums in the Spotify dataset. The most represented album is "Alternative Christmas 2022," with around 200 songs. It is followed by "Feliz Cumpleaños con Perreo" with approximately 150 songs. Other notable albums include "Halloween con perro," "Halloween Party 2022," and "The Complete Hank Williams," all with a considerable number of songs.



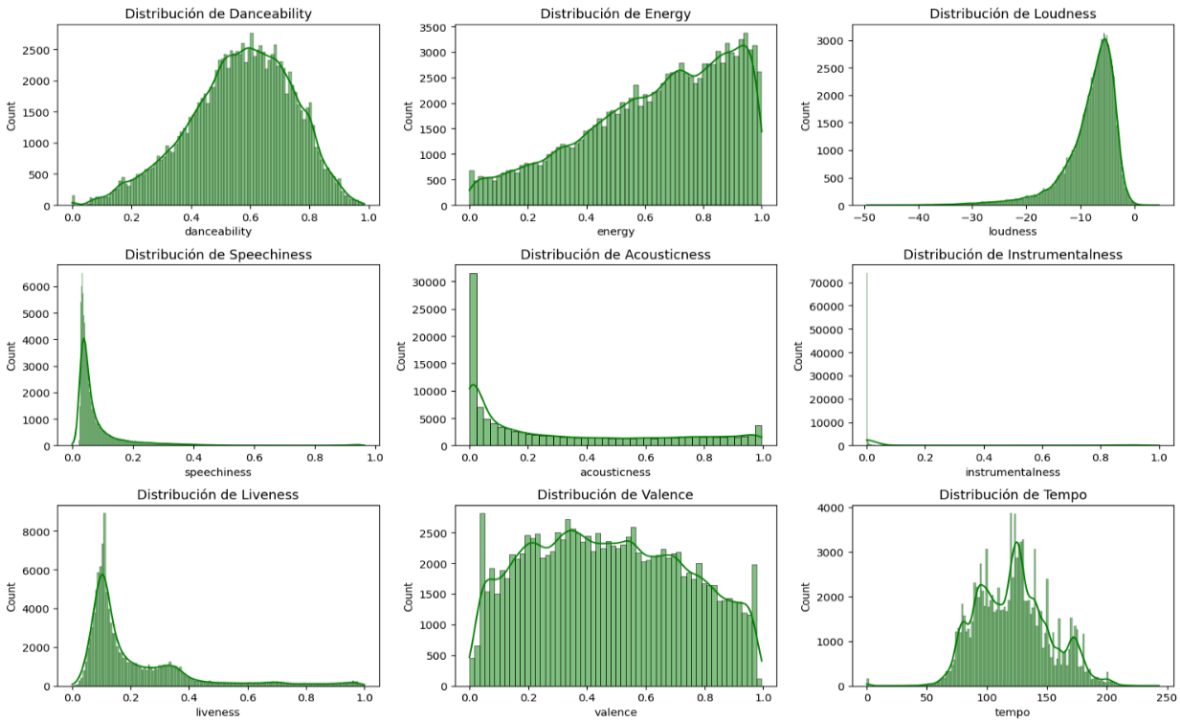
This heatmap shows the correlations between different audio characteristics of songs in the Spotify dataset.

Here are some important relationships:

* There is a strong positive correlation between energy and loudness (0.76), indicating that more energetic songs tend to be louder.

* Danceability and valence have a moderate correlation (0.48), suggesting that more danceable songs tend to have a more positive or upbeat tone.

* Acousticness has a strong negative correlation with energy (-0.73) and loudness (-0.59), implying that more acoustic songs tend to be less energetic and softer.



This set of graphs shows the distributions of various audio features in the Spotify dataset.

Here are some key takeaways:

1. Danceability: Most songs have a danceability value between 0.5 and 0.8, suggesting that many songs are quite danceable.

2. Energy: The energy of the songs is fairly evenly distributed, with a slight bias toward high values, implying that the songs tend to be energetic.

3. Loudness: Most songs have a loudness value between -10 and 0 dB, indicating that they are generally quite loud.

4. Speechiness: Most songs have very low speechiness values, meaning that most are primarily musical rather than spoken.

5. Acousticness: A large portion of the songs have a low acousticness value, suggesting that many songs are not acoustic.

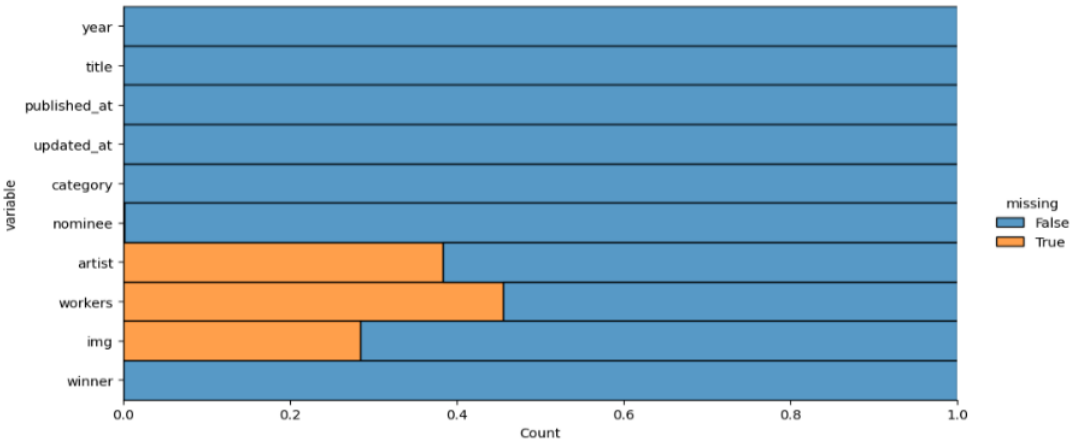
6. Instrumentalness: Most songs have a value close to 0, indicating that most songs have lyrics and are not instrumental.

7. Liveness: Most songs have low liveness values, suggesting that most were not recorded live.

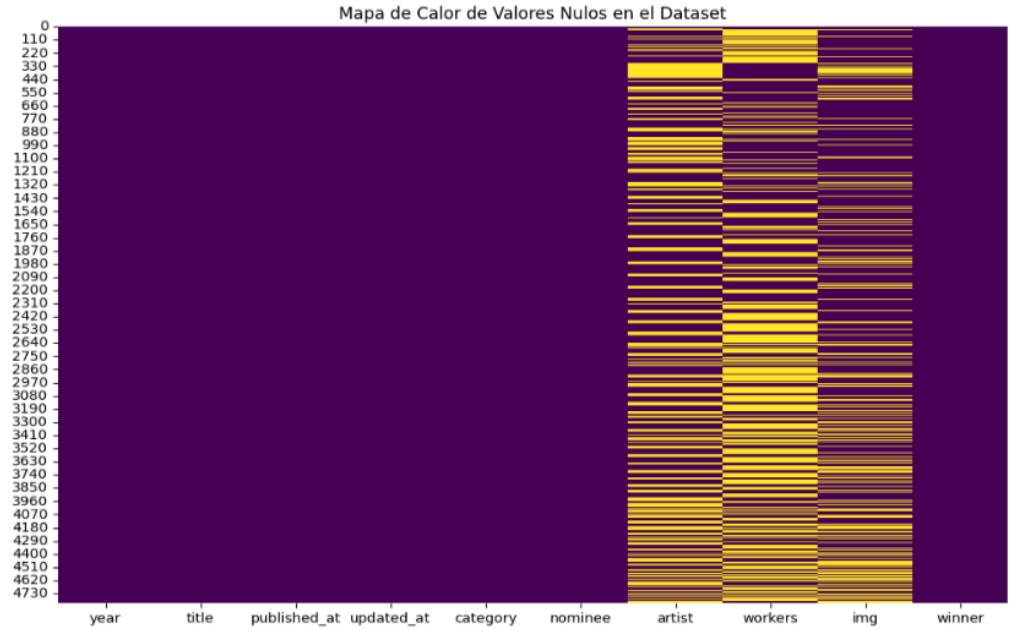
8. Valence: Valence is fairly evenly distributed, suggesting a mix of songs with positive and negative emotions.

9. Tempo: Tempo shows a bimodal distribution, with peaks around 100 and 120 BPM, which is typical for moderate to fast-paced songs.

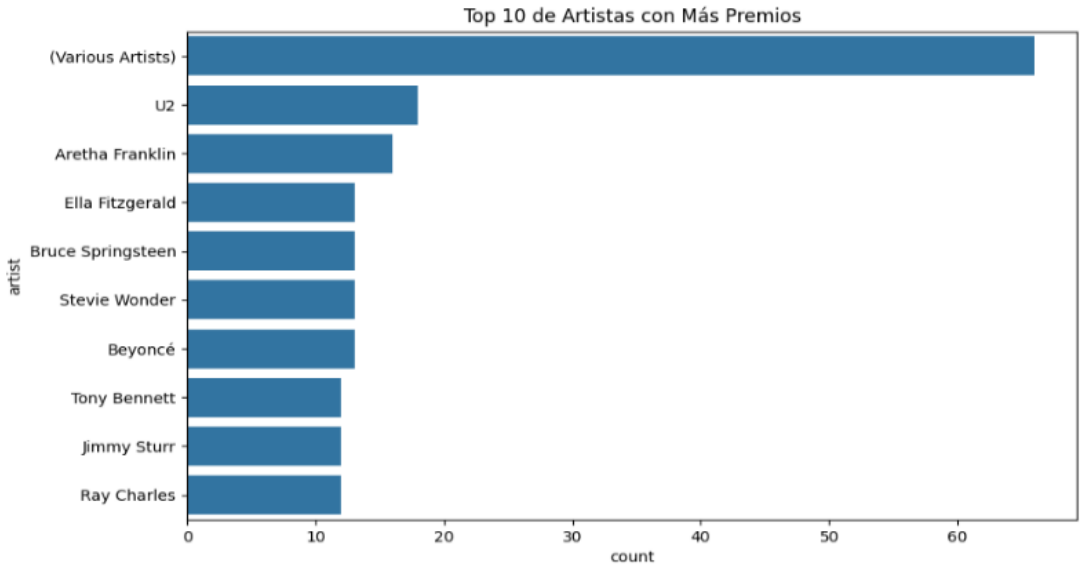
Eda visualizations of Grammy dataset



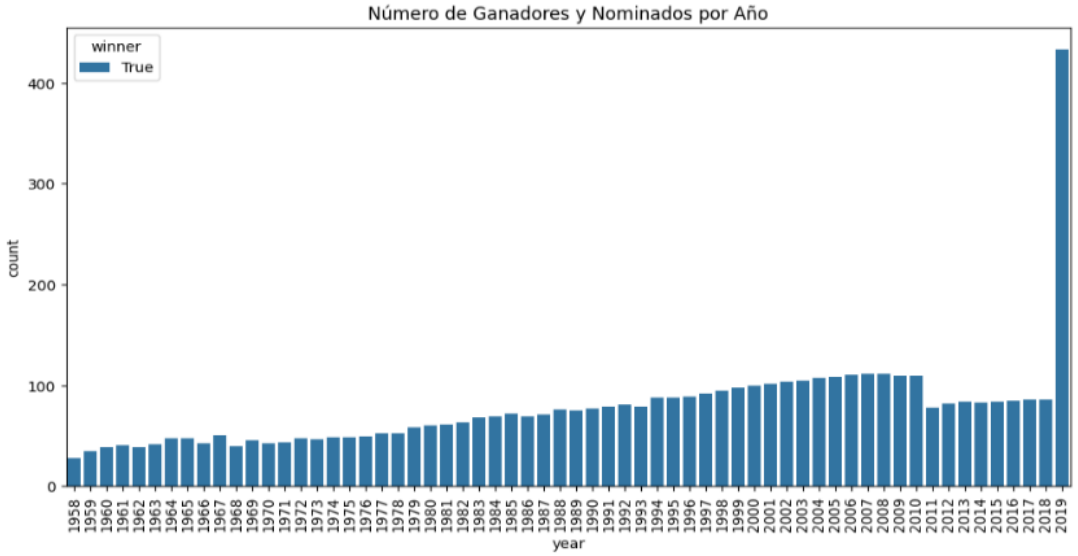
In this chart we can see that the columns `year`, `title`, `published_at`, `updated_at`, `category`, and `winner` are completely filled, while the columns `artist`, `workers`, and `img` have a significant amount of missing data, with approximately 40%, 50%, and 30% of null values respectively.



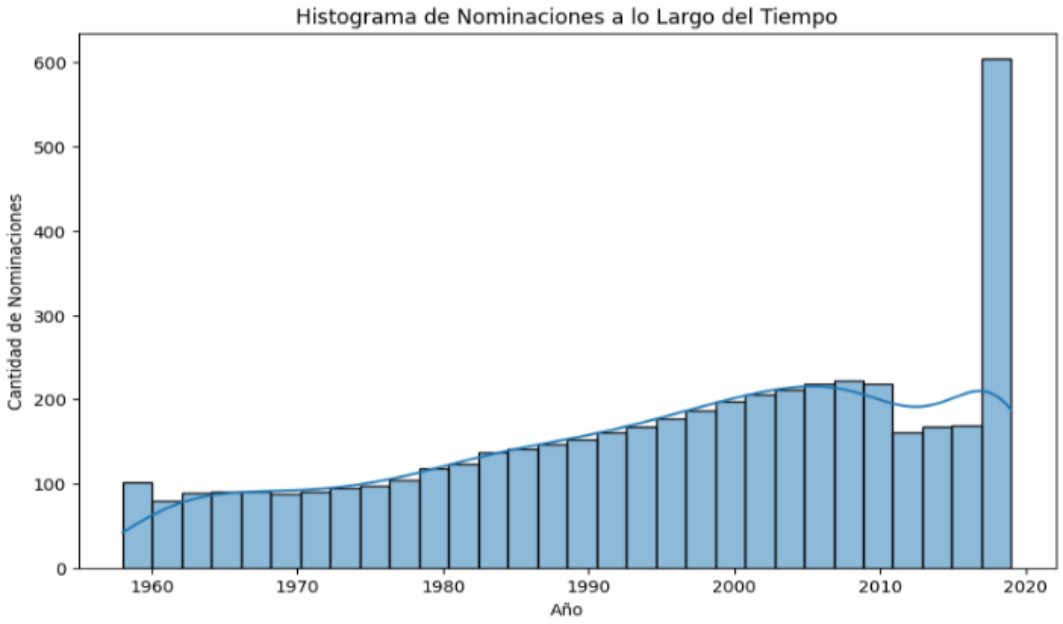
The null value heatmap shows that columns such as `year`, `title`, `category`, and `winner` have no missing values, while columns `artist`, `workers`, and `img` have a significant amount of null values. The `workers` column is the most affected, with approximately 50% missing data, followed by `artist` and `img`.



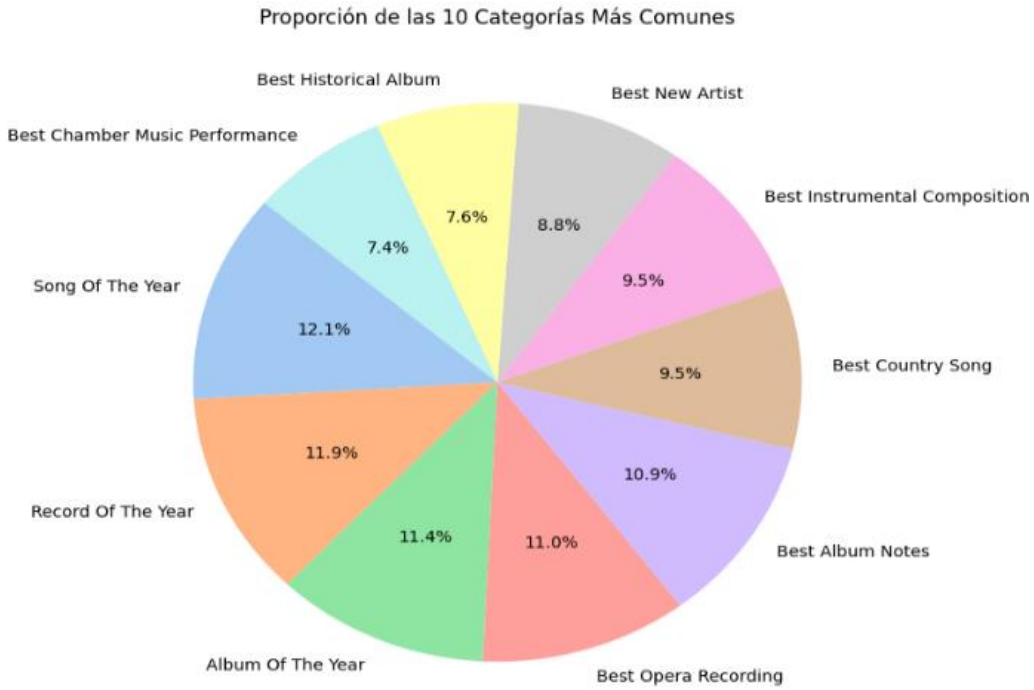
The chart shows the top 10 artists with the most Grammy wins. In first place is "Various Artists," which could refer to collaborations or compilations of several artists that have won a significant number of awards, far outnumbering the others. Among individual artists, U2 tops the list, followed by Aretha Franklin, Ella Fitzgerald, and other big names such as Bruce Springsteen, Stevie Wonder, and Beyoncé.



The chart shows the number of winners per year at the Grammy Awards from 1958 to 2019. There is a general upward trend in the number of winners over the years, with the most notable increase beginning in the 1980s. The number of winners seems to stabilize in the 2000s, with slight fluctuations. However, in 2019 there is a significant jump, which could be due to an increase in the number of categories or special awards that year.



This histogram shows the number of Grammy Award nominations over time. Since the 1960s, there has been a gradual increase in nominations, with more marked growth beginning in the 1980s and a notable peak in 2020.



The pie chart shows the proportion of the 10 most common categories at the Grammy Awards. "Song of the Year" is the most represented category, with 12.1% of the total, followed by "Record of the Year" and "Album of the Year," with shares close to 11%. Other significant categories include "Best New Artist" and "Best Instrumental Composition," each with around 9-10%. The least frequent categories in this group are "Best Chamber Music Performance" and "Best Historical Album."

Now we are going to carry out the process with airflow.

After defining the functions to do in Airflow, we wrote the code for each one and for the dag.

GRAMMY LIST

We read the data from the postgres db

We eliminate duplicates.

We eliminate the columns img, workers, updated_at and title.

We eliminate repeated data in nominee and category

We convert boolean values of the "winner" column to 1 and 0.

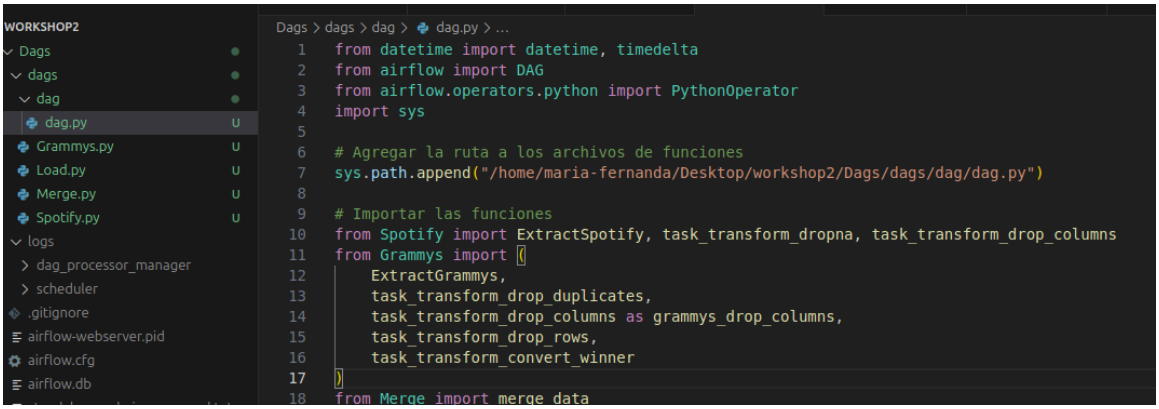
SPOTIFY LIST.

We read the data from the csv

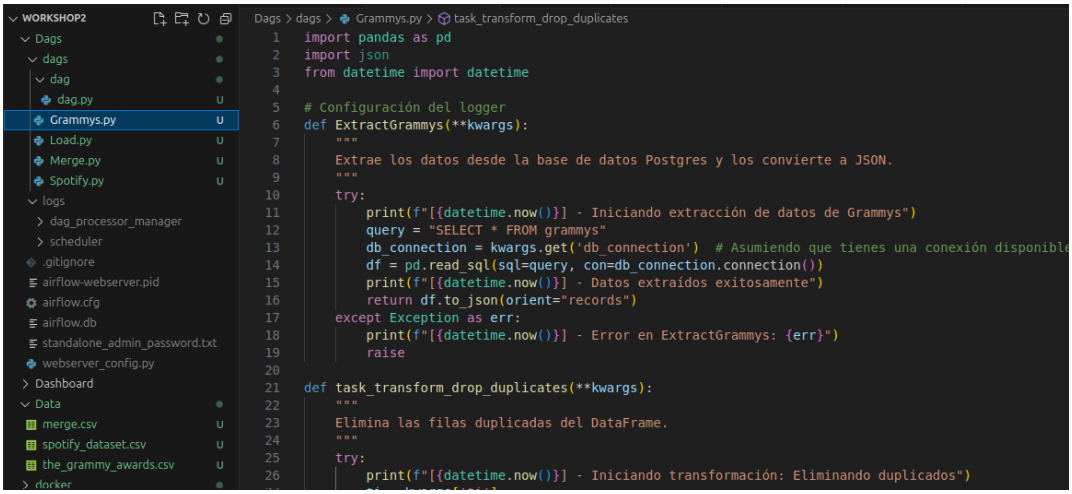
We eliminate null data.

Transformation: we eliminate the column Unnamed: 0 and track_id.

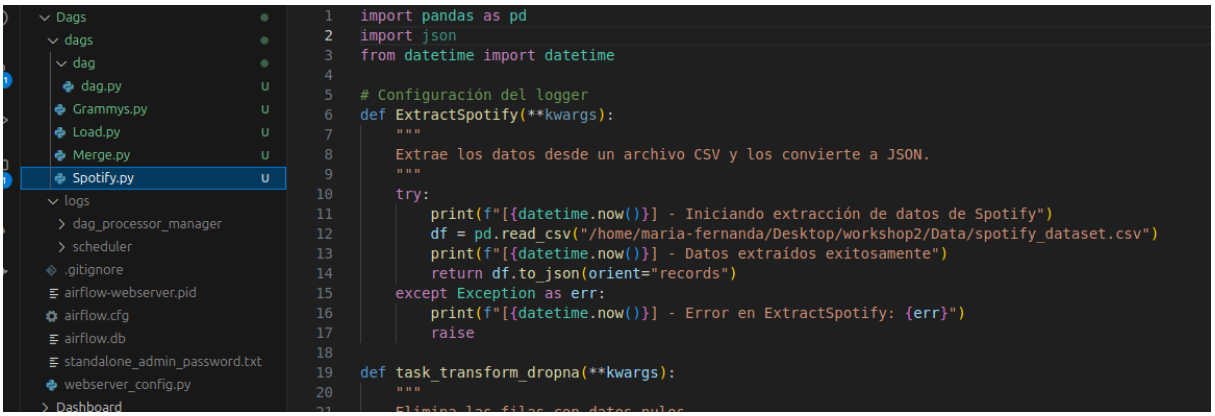
Dag.py



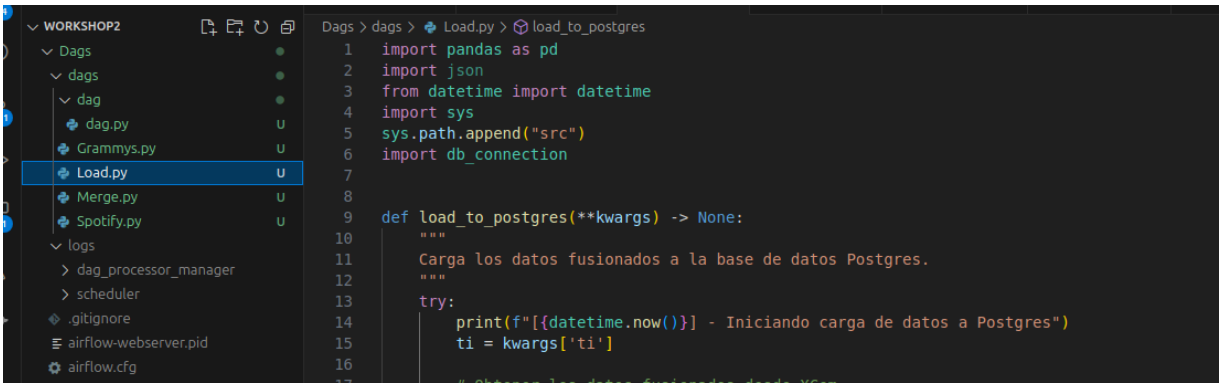
Grammys.py



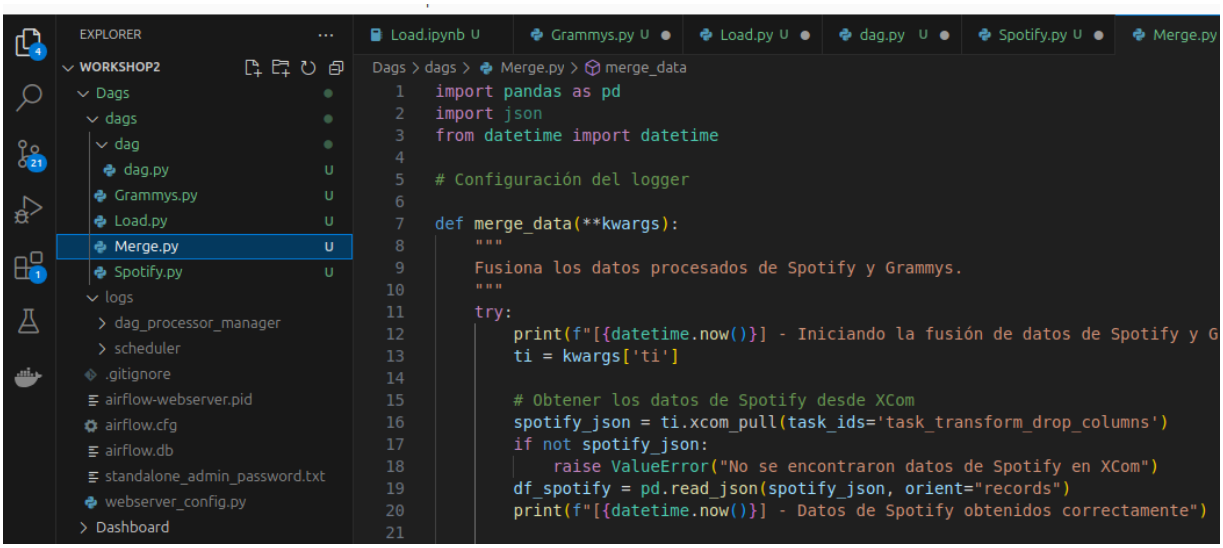
Spotify.py



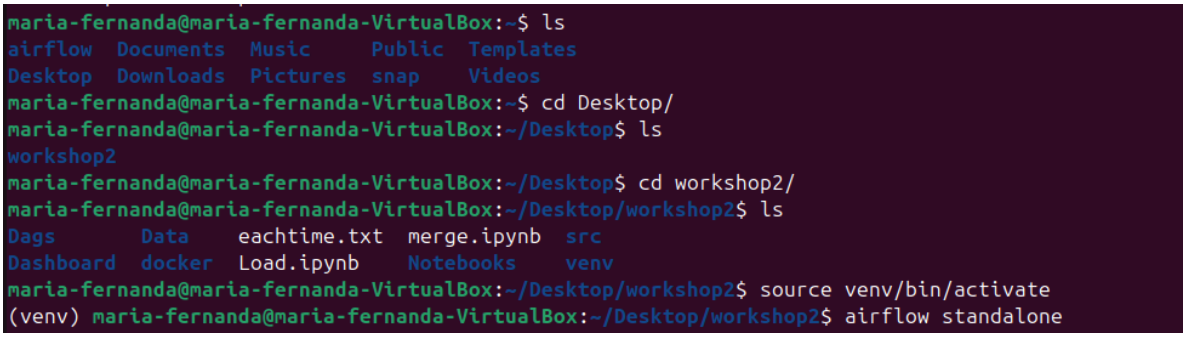
Load.py



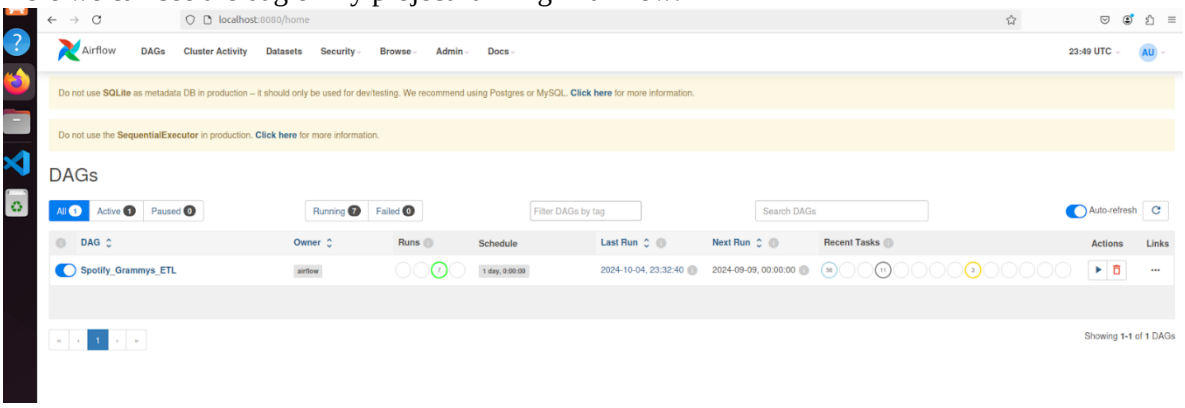
Merge.py



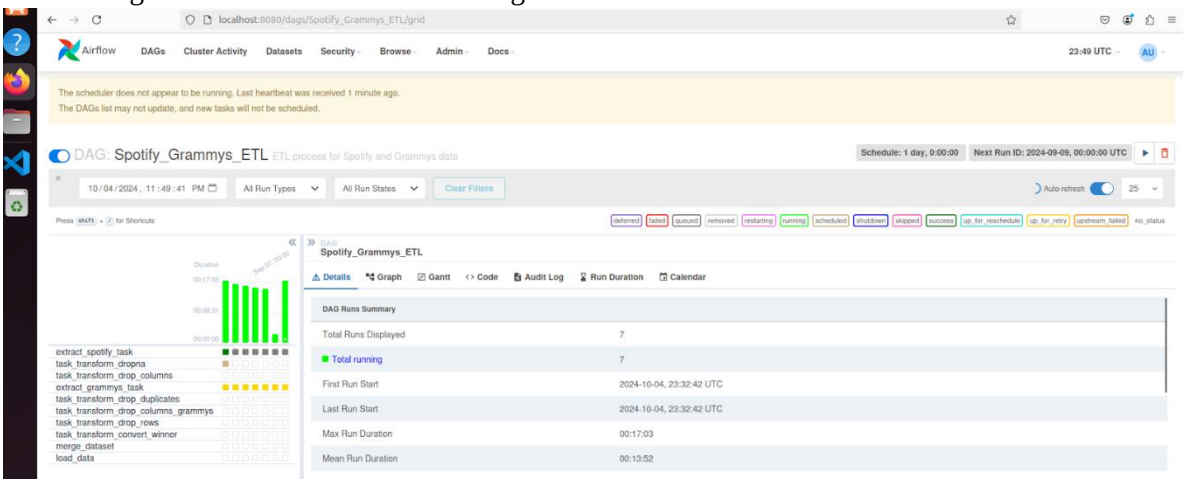
Run the airflow.



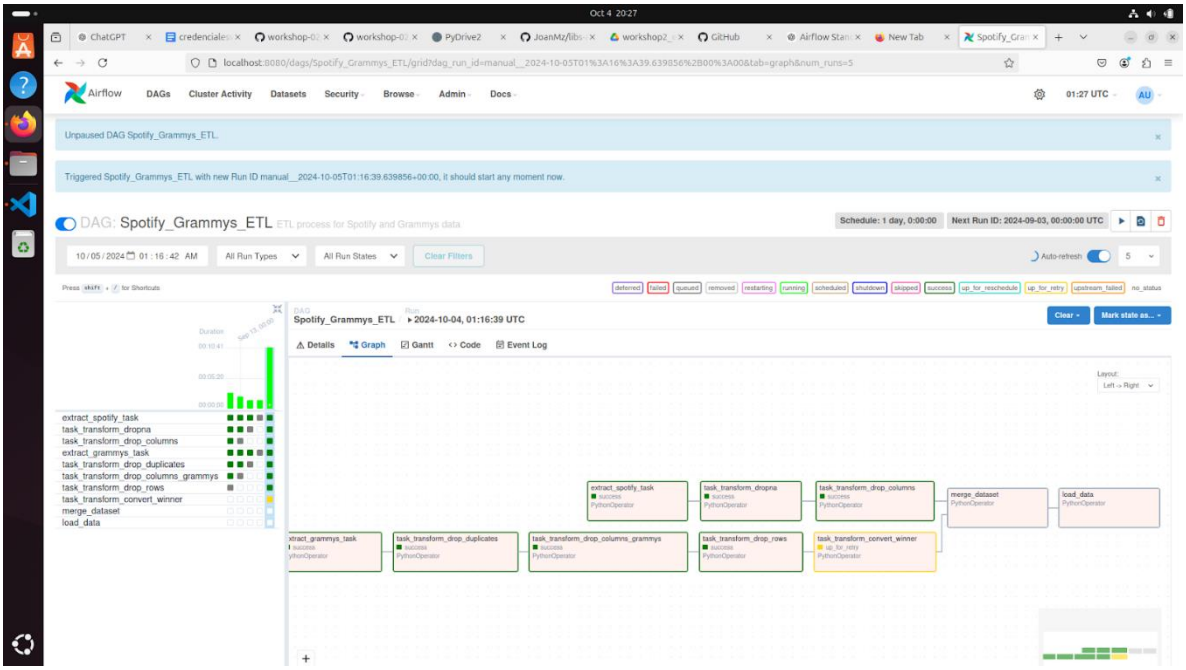
Here we can see the dag of my project running in airflow.



In this image we can see some tasks running.

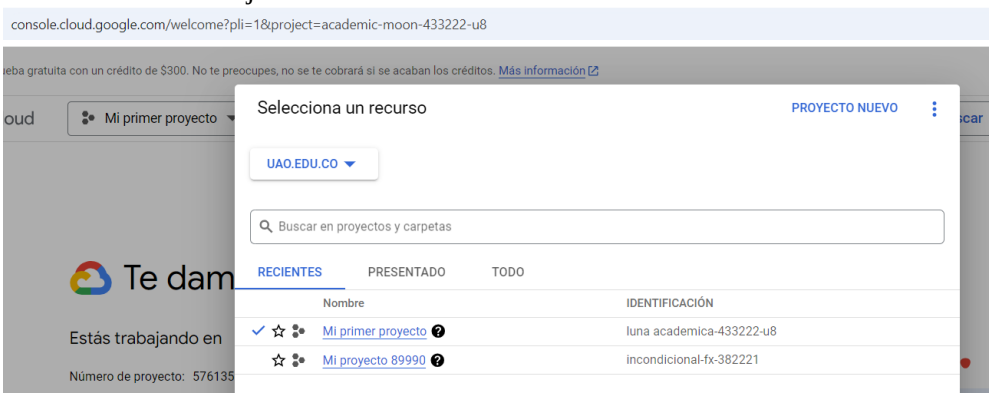


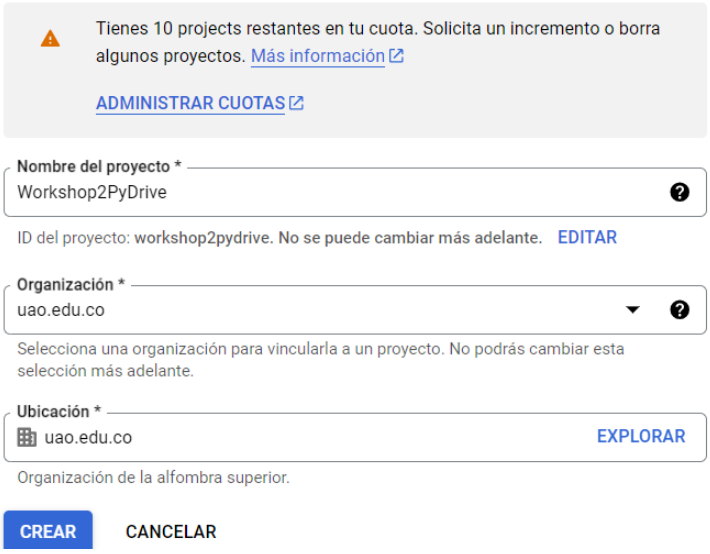
Finally, we can see most of the tasks running perfectly, and we can also see that the connection to the Postgres database was made correctly, since the Grammys dataset runs perfectly from the first task.



GET ON THE DRIVE

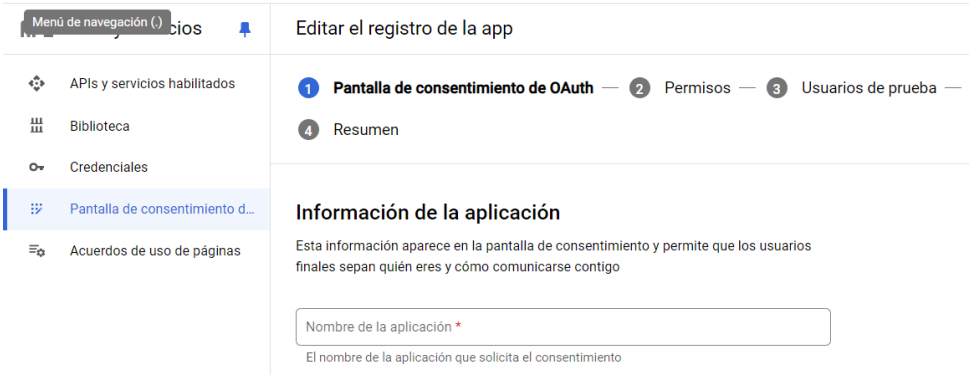
We create a new Project



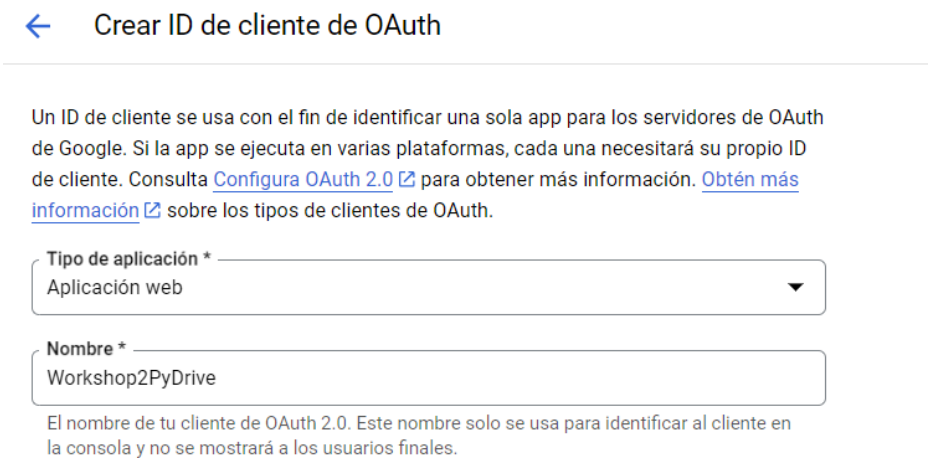


We enable the Google Drive API for our project.

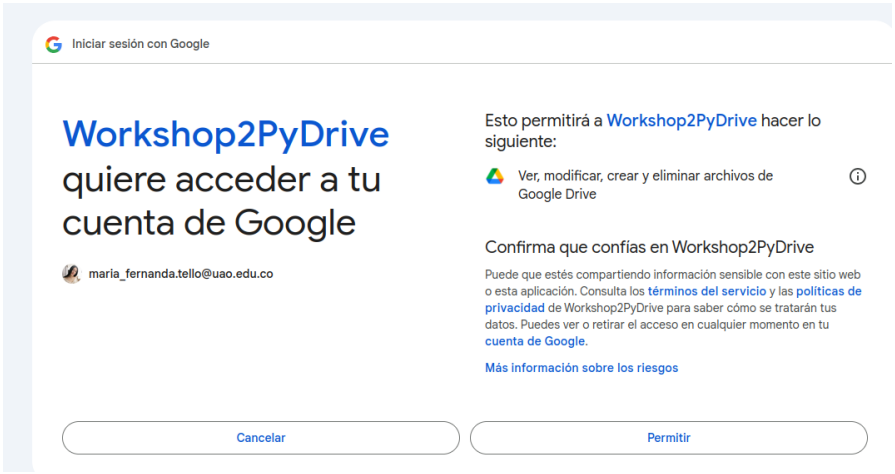
Configure the consent screen.

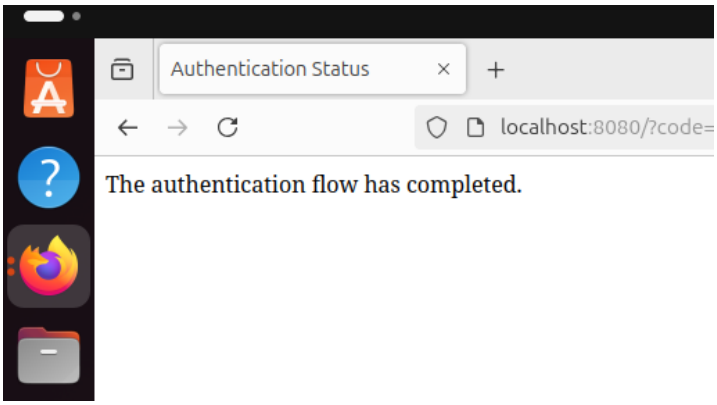


We create the credentials.



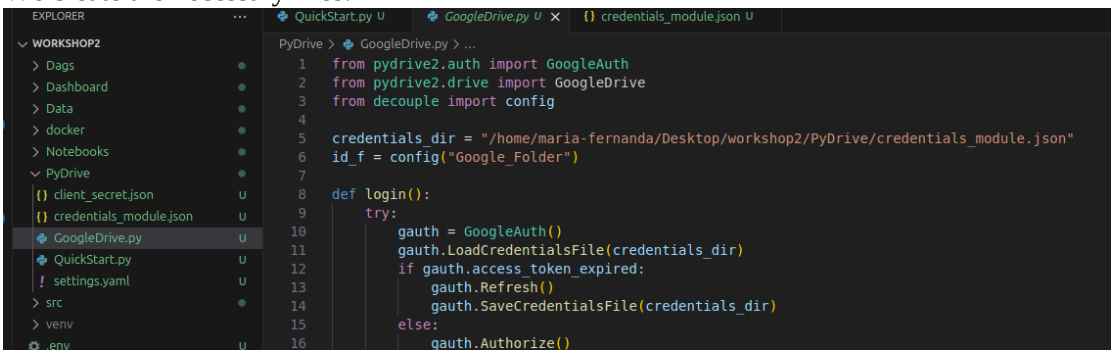
Credentials created.






```
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/PyDrive$ python3 QuickStart.py
/home/maria-fernanda/Desktop/workshop2/venv/lib/python3.12/site-packages/oauth2client/_helpers.py:
y
  warnings.warn(_MISSING_FILE_MESSAGE.format(filename))
Your browser has been opened to visit:

    https://accounts.google.com/o/oauth2/auth?client_id=411865893581-a8mvt5umrg0gr89ts349oljecbl4c
e=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive&access_type=offline&response_type=code&approval_

Authentication successful.
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/PyDrive$
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/PyDrive$
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2/PyDrive$
```

We create the necessary files.



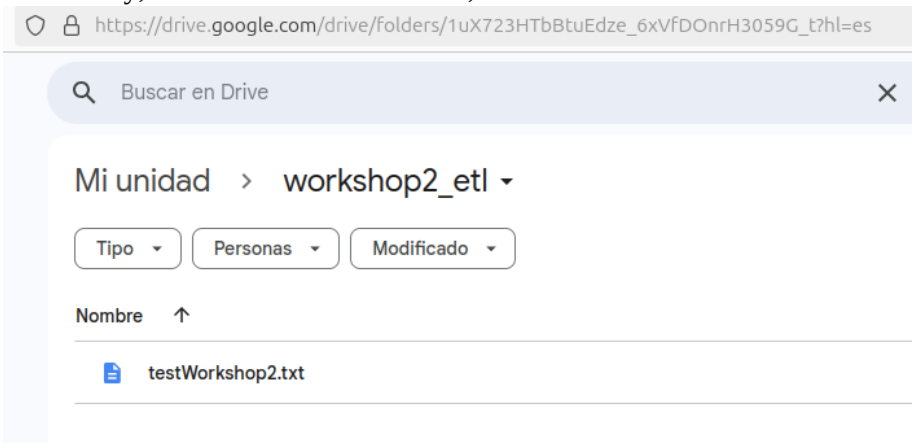
We configure the GoogleDrive.py file, and perform a test to make sure it works when uploading files to my Google Drive.

```
    file["title"] = path_file.split("/")[-1]
    file.SetContentFile(path_file)
    file.Upload()



if __name__ == "__main__":
    Create_file("testWorkshop2.txt", "holi holi", "1uX723HTbBtuEdze_6xVfDOnrH3059G_t")
    Upload_file("../Data/merge.csv", "1uX723HTbBtuEdze_6xVfDOnrH3059G_t")
    pass
```

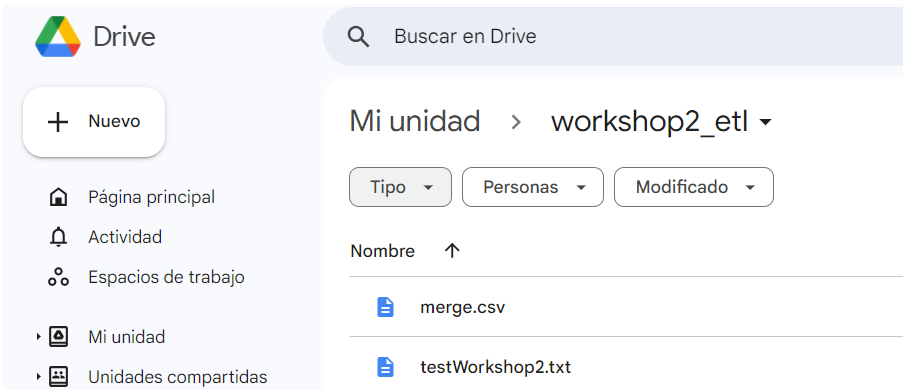
Here we can see it running.

```
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$ /home/maria-fernanda/Desktop/workshop2/venv/bin/python
/home/maria-fernanda/Desktop/workshop2/PyDrive/GoogleDrive.py
(venv) maria-fernanda@maria-fernanda-VirtualBox:~/Desktop/workshop2$
```

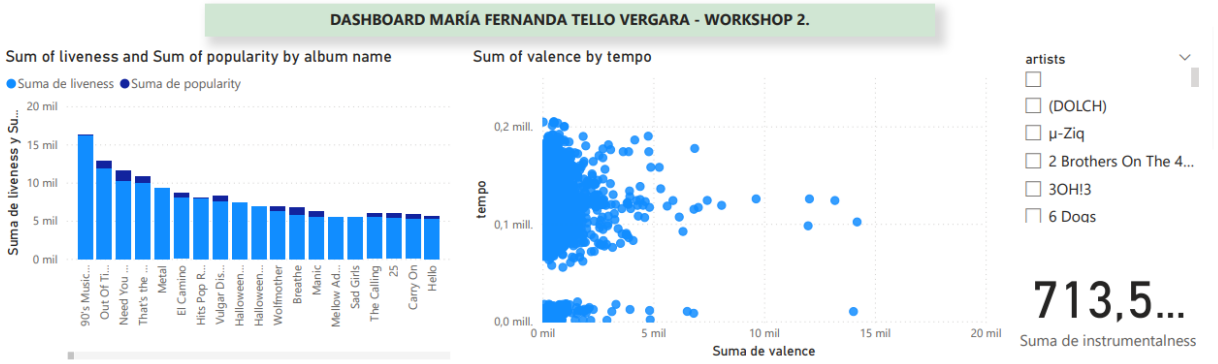
And finally, we observe the created file, in the Drive.

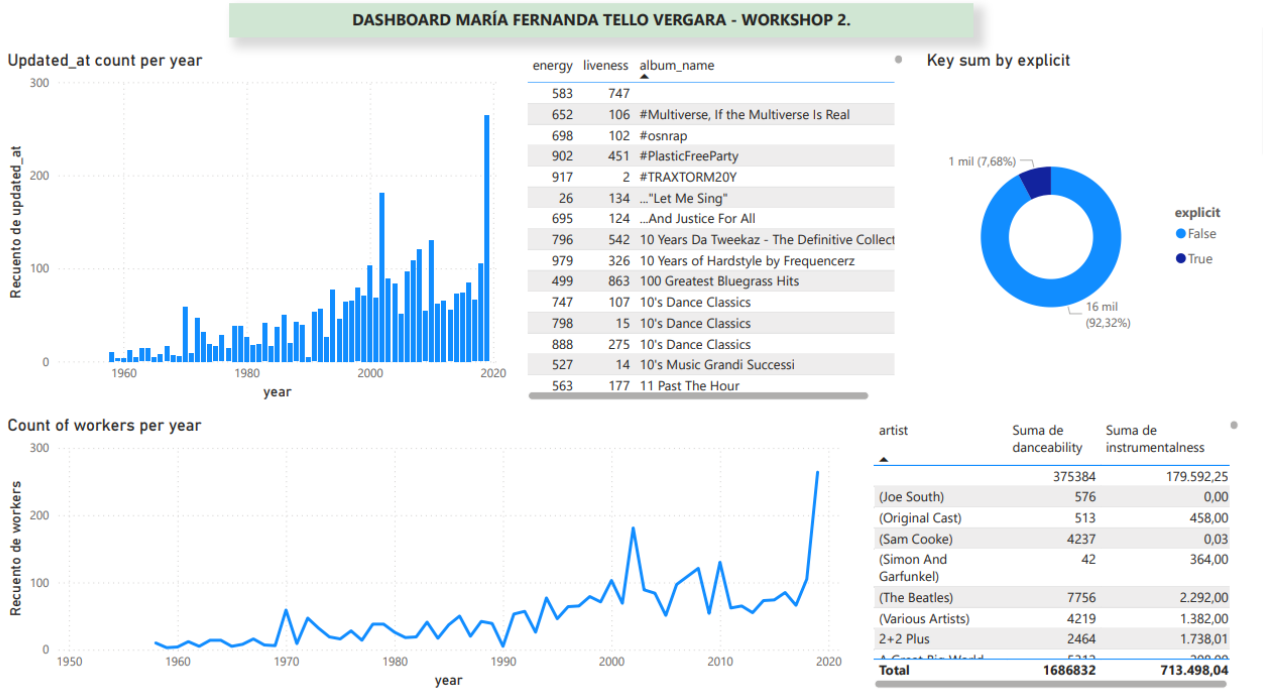


Furthermore, in the following image we can see that it actually loads the merge.csv file, in the same folder.



VISUALIZATIONS





Here are some of the questions we'll be answering in the Power BI dashboard:

- How much has the number of updates been per year?
- How has the number of workers changed over the years?
- Which albums have the highest sum of liveness and popularity?
- What proportion of the music is explicit or not?
- How are tempo and valence distributed between different tracks?
- Which workers contributed to the songs with the highest liveness and popularity?
- What are the most common musical genres in the tracks?
- Which artists have the highest sum of "danceability" and instrumentality?

Finally, we uploaded all the folders to the GitHub repository!