II.2414 - BASES DE DONNEES ET BIG DATA

REPORT

BIG DATA PROJECT

Sara Fernández Malvido sara.fernandez-malvido@eleve.isep.fr

Mario Aroca Paéz mario.aroca-paez@eleve.isep.fr

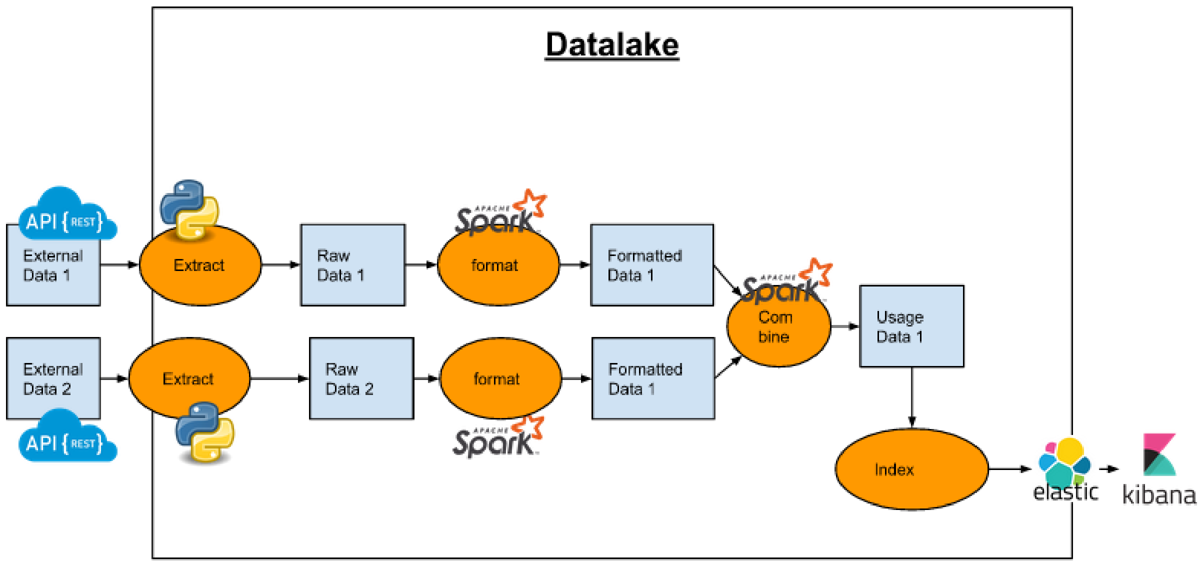
Table of contents

1. Introduction

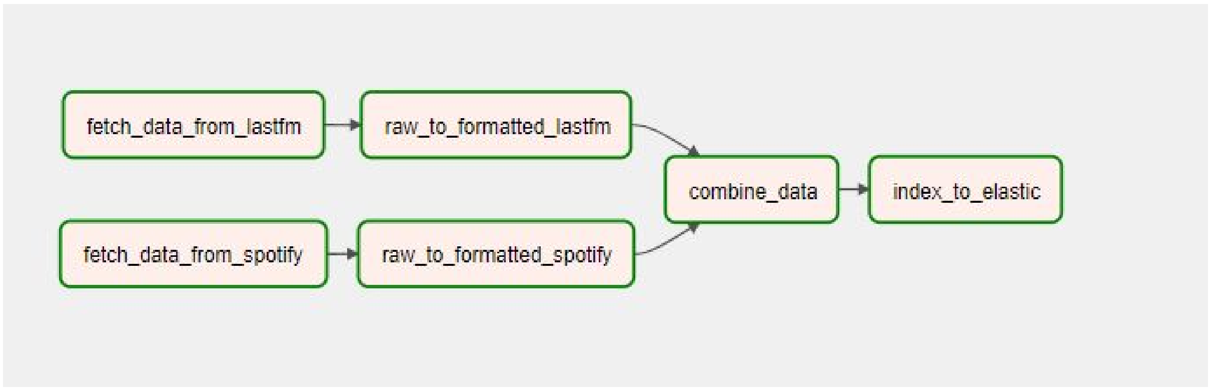
The big data project described in this report focuses on creating a comprehensive end-to-end data architecture for music analysis. This project uses data from various sources such as the Last.fm API and the Spotify API to perform analysis of music trends and artist information. The data pipeline is orchestrated using Airflow and the data is stored locally in the project's file system. A pipeline includes processes such as ingesting, formatting, joining, and indexing data that lead to the generation of insightful results for further analysis. This report provides an overview of the project's architecture, data sources, data pipelines, and the ultimate goal of analyzing music-related data.

2. Architecture

The goal of this Big Data project is to create a simple end-to-end data architecture, including data ingestion, data transformation and data transformation.
Here there is a description of the final architecture of the project:



All the different jobs for this data lake are orchestrated using airflow, generating different DAGS for each operation resulting in a DAG with the following structure:



And the storage of all the data is done in the local filesystem where the project is running.

2.1. Theme

The theme of this project is Music. Some analysis will be performed with different data regarding music and current trends nowadays.

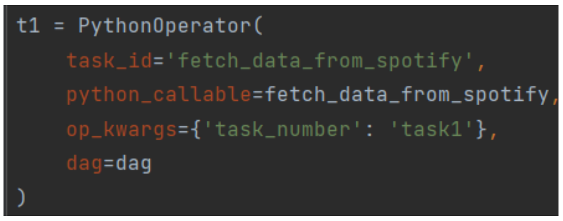
2.2. Datasources

Two different data sources have been used for this project. The first one is the official Last.fm API where data about different artist was retrieved, this data contains the artist name, his lastfm URL, number of listeners, short biography, list of similar artists, list of top songs and a Yes or No saying if the artist is currently on tour or not.

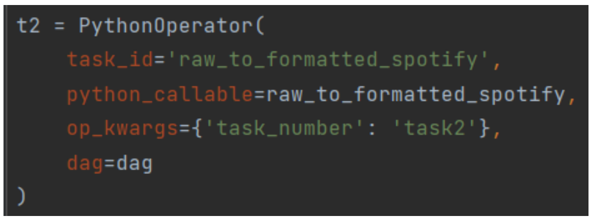
The second data source is the official Spotify API where data from two different spotify playlist was retrieved. First from the Top 50: Spain playlist where the fifty most listened songs in Spain can be found and the Top 50: Global playlist which has the same but for the whole word. The data extracted from here contains for each song a Track Name, Artist, number from one to one hundred showing Popularity and the spotify URL. Note that this retrieval of information is done for two different entities is the data lake, one for Spain and the other one for the Global

3. Data Pipeline

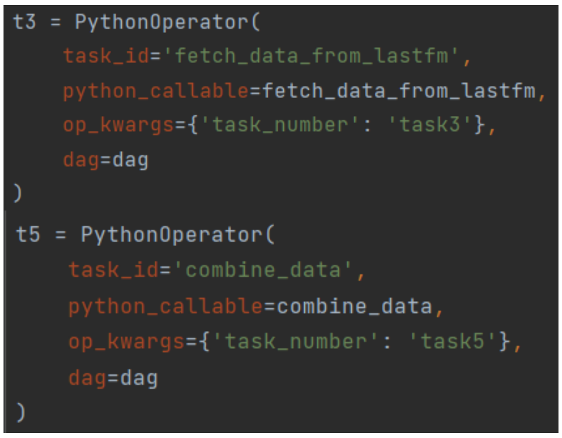
The whole data pipeline is created as an Airflow DAG in order to orchestrate all the jobs easier, this is done using python. Six tasks will be created: data fetcher from Spotify, data fetcher from LastFM, raw to formatted from Spotify, raw to formatted from LastFM, the combination of the data and the index to elastic.
The tasks will be executed in the following order: First we fetch the data and once the data is fetched, we will convert this raw data to formatted data. Once these steps have been done for both sources, the combination of the data is performed. Finally, once the previous step is done, it is indexed to elastic.



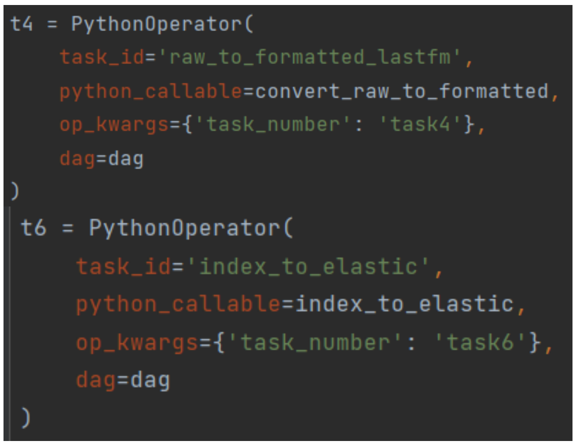
```
t1 = PythonOperator(
    task_id='fetch_data_from_spotify',
    python_callable=fetch_data_from_spotify,
    op_kwargs={'task_number': 'task1'},
    dag=dag
)
```



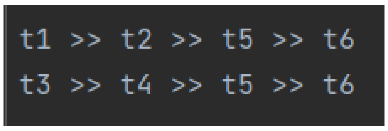
```
t2 = PythonOperator(
    task_id='raw_to_formatted_spotify',
    python_callable=raw_to_formatted_spotify,
    op_kwargs={'task_number': 'task2'},
    dag=dag
)
```



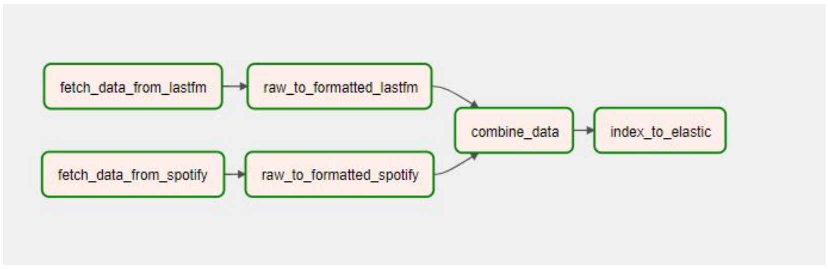
```
t3 = PythonOperator(
    task_id='fetch_data_from_lastfm',
    python_callable=fetch_data_from_lastfm,
    op_kwargs={'task_number': 'task3'},
    dag=dag
)
t5 = PythonOperator(
    task_id='combine_data',
    python_callable=combine_data,
    op_kwargs={'task_number': 'task5'},
    dag=dag
)
```



```
t4 = PythonOperator(
    task_id='raw_to_formatted_lastfm',
    python_callable=convert_raw_to_formatted,
    op_kwargs={'task_number': 'task4'},
    dag=dag
)
t6 = PythonOperator(
    task_id='index_to_elastic',
    python_callable=index_to_elastic,
    op_kwargs={'task_number': 'task6'},
    dag=dag
)
```



```
t1 >> t2 >> t5 >> t6
t3 >> t4 >> t5 >> t6
```



3.1. Ingestion

For the Ingestion, first when the fetch_data_from_lastfm DAG is triggered it runs a script named data_fetcher_lastfm.py that makes an API request to the lastfm API and generates a JSON file with the already specified data from the API. It gets that data for the top 100 artists in lastfm which is constantly changing.

Then when the fetch_data_from_spotify DAG is triggered it runs a script named data_fetcher_spotify.py this executes two different scripts, one for the TOP 50 global playlist and the other one for the TOP 50 spain playlist and as before it stores all the data in two different JSON files.

It is worth noticing that a naming convention was defined so the project has a clean structure:

raw/
raw/lastfm/
raw/lastfm/Artist
raw/lastfm/Artist/date
raw/lastfm/Artist/date/lastfm.json

raw/

raw/spotify/
raw/spotify/TrackListGlobal/
raw/spotify/TrackListGlobal/date/
raw/spotify/TrackListGlobal/date/tracklistglobal.json
raw/spotify/TrackListSpain
raw/spotify/TrackListSpain/date/
raw/spotify/TrackListSpain/date/tracklistspain.json

3.2. Formatting

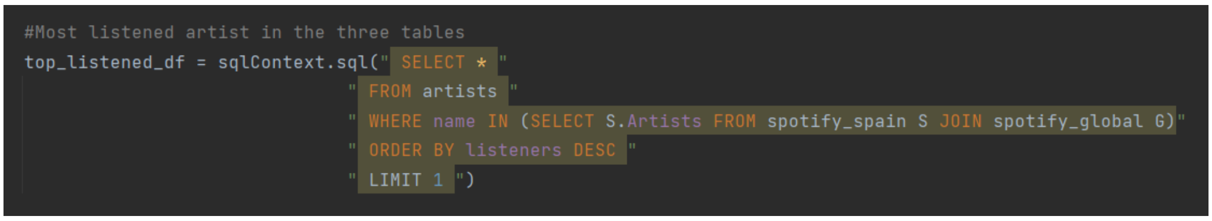
The formatting is done once the raw_to_formatted_lastfm and raw_to_formatted_spotify DAGS are triggered; they run the raw_to_fmt_lastfm.py and raw_to_fmt_spotify.py files. Theses scripts take the JSON files and using the pandas library it converts them to parquet format and store them in new directories:

formatted/
formatted/lastfm/
formatted/lastfm/Artist
formatted/lastfm/Artist/date
formatted/lastfm/Artist/date/lastfm.parquet

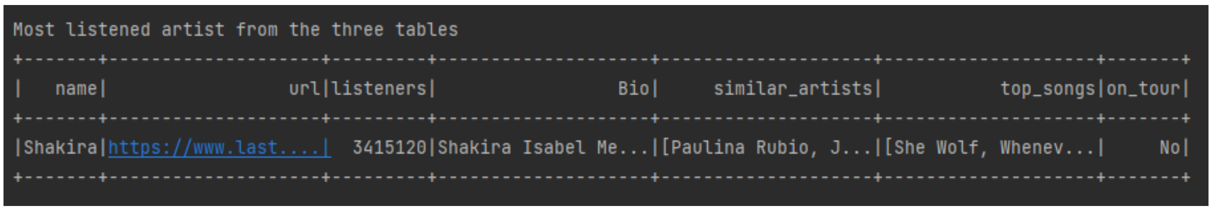
formatted/
formatted/spotify/
formatted/spotify/TrackListGlobal/
formatted/spotify/TrackListGlobal/date/
formatted/spotify/TrackListGlobal/date/tracklistglobal.parquet
formatted/spotify/TrackListSpain
formatted/spotify/TrackListSpain/date/
formatted/spotify/TrackListSpain/date/tracklistspain.parquet

3.3. Combination

Once the data is formatted it is ready to be combined, therefore once the combine_data DAG is triggered it executes the combine_data.py file that takes this data and combines it using SQL statements. The combination operations are the most listened artist in the three tables:

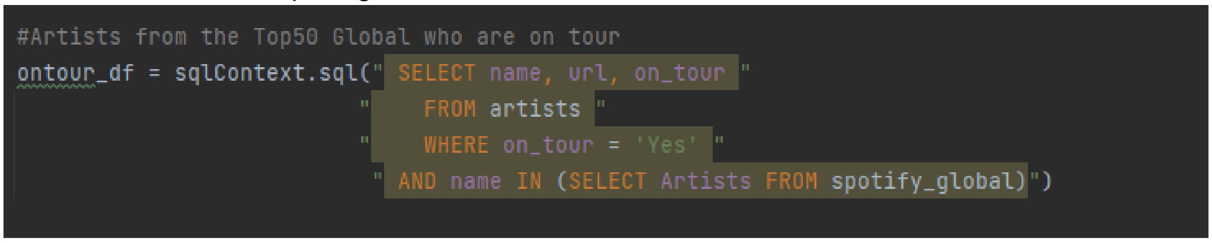


```
#Most listened artist in the three tables
top_listened_df = sqlContext.sql(" SELECT * "
                                 " FROM artists "
                                 " WHERE name IN (SELECT S.Artists FROM spotify_spain S JOIN spotify_global G)"
                                 " ORDER BY listeners DESC "
                                 " LIMIT 1 ")
```

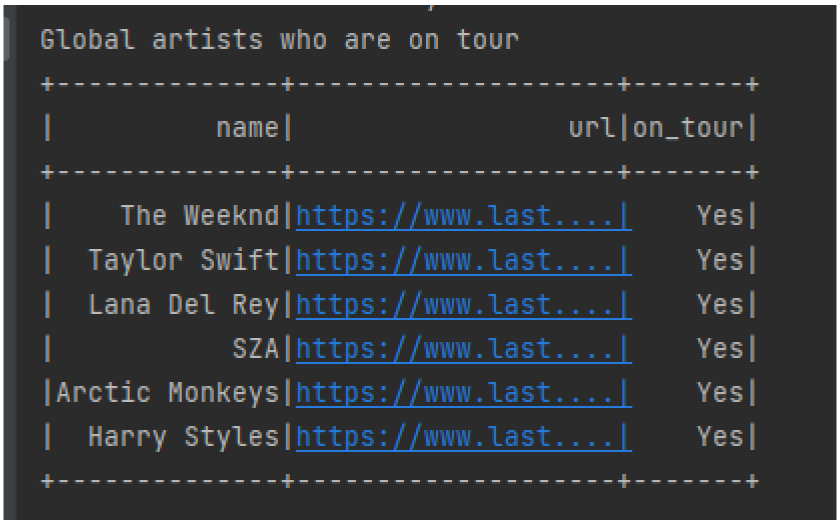


```
Most listened artist from the three tables
+-------+--------------------+---------+--------------------+--------------------+--------------------+-------+
|   name|                 url|listeners|                 Bio|     similar_artists|           top_songs|on_tour|
+-------+--------------------+---------+--------------------+--------------------+--------------------+-------+
|Shakira|https://www.last....|  3415120|Shakira Isabel Me...|[Paulina Rubio, J...|[She Wolf, Whenev...|     No|
+-------+--------------------+---------+--------------------+--------------------+--------------------+-------+
```

The artists from the Top 50 global who are on tour:

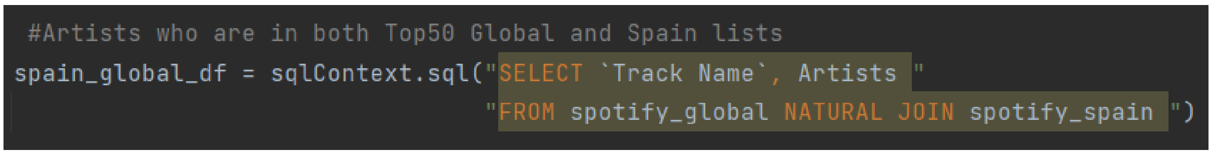


```
#Artists from the Top50 Global who are on tour
ontour_df = sqlContext.sql(" SELECT name, url, on_tour "
                          "     FROM artists "
                          "     WHERE on_tour = 'Yes' "
                           " AND name IN (SELECT Artists FROM spotify_global)")
```

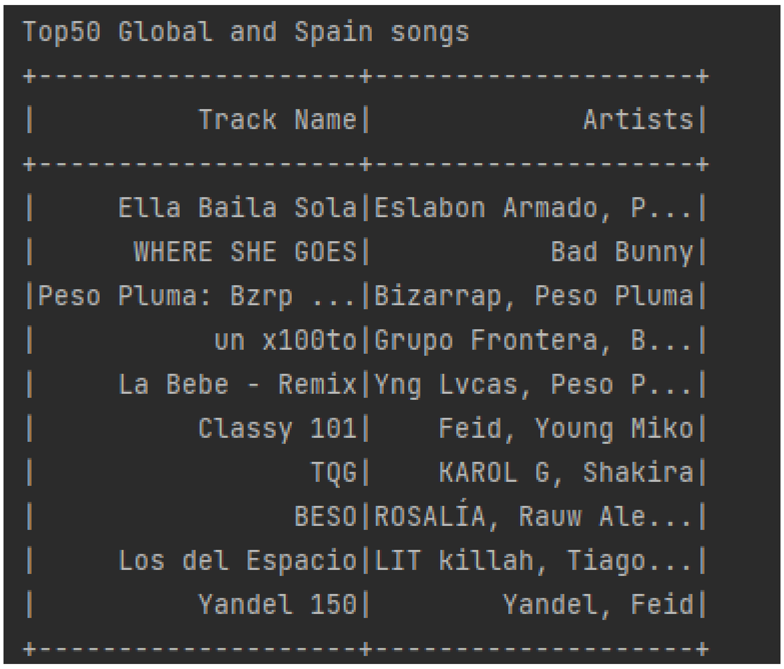


```
Global artists who are on tour
+--------------+--------------------+-------+
|          name|                 url|on_tour|
+--------------+--------------------+-------+
|    The Weeknd|https://www.last....|    Yes|
|  Taylor Swift|https://www.last....|    Yes|
|  Lana Del Rey|https://www.last....|    Yes|
|           SZA|https://www.last....|    Yes|
|Arctic Monkeys|https://www.last....|    Yes|
|  Harry Styles|https://www.last....|    Yes|
+--------------+--------------------+-------+
```

And the artists who are in both the Top 50 Global and Top 50 Spain lists:



```
#Artists who are in both Top50 Global and Spain lists
spain_global_df = sqlContext.sql("SELECT `Track Name`, Artists "
                                 "FROM spotify_global NATURAL JOIN spotify_spain ")
```



```
Top50 Global and Spain songs
+--------------------+--------------------+
|          Track Name|             Artists|
+--------------------+--------------------+
|     Ella Baila Sola|Eslabon Armado, P...|
|      WHERE SHE GOES|           Bad Bunny|
|Peso Pluma: Bzrp ...|Bizarrap, Peso Pluma|
|            un x100to|Grupo Frontera, B...|
|      La Bebe - Remix|Yng Lvcas, Peso P...|
|          Classy 101|    Feid, Young Miko|
|                  TQG|    KAROL G, Shakira|
|                 BESO|ROSALÍA, Rauw Ale...|
|     Los del Espacio|LIT killah, Tiago...|
|          Yandel 150|        Yandel, Feid|
+--------------------+--------------------+
```

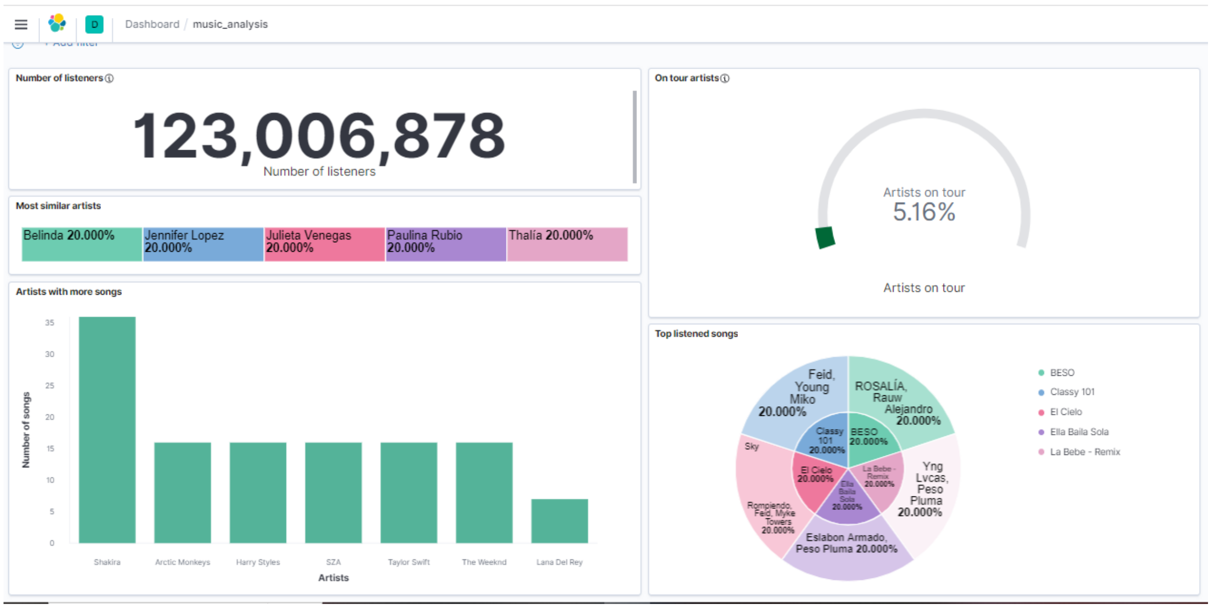
After this new files in the data lake are created:

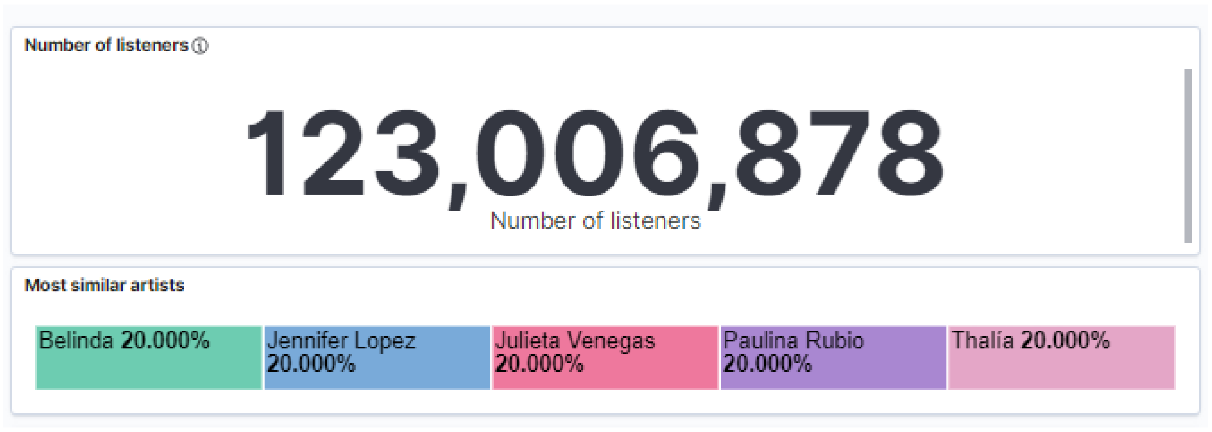
```
usage/
usage/musicAnalysis/
usage/musicAnalysis/GlobalOnTour/
usage/musicAnalysis/GlobalOnTour/date/global_ontour.parquet
usage/musicAnalysis/SongsSpainGlobal/
usage/musicAnalysis/SongsSpainGlobal/date/
usage/musicAnalysis/SongsSpainGlobal/date/top_spain_global.parquet
usage/musicAnalysis/TopArtistListened/
usage/musicAnalysis/TopArtistListened/date/
usage/musicAnalysis/TopArtistListened/date/top_artist.parquet
```

3.4. Indexing

Finally after the data is combined the index_to_elastic DAG is triggered, this runs the index_to_elastic.py that takes the formatted data, connects to elastic wich has to be running at the same time, sets the index name to music_analysis, turns the combined data into JSON again because elastic does not accept parquet and indexes this data into the music_analysis index.

When all the data is indexed using elasticsearch the results are shown in Kibana. Accessing kibana we can see the total number of listeners in the data, percentage of artists on tour, the artists with more songs and the top listened songs with their artist.





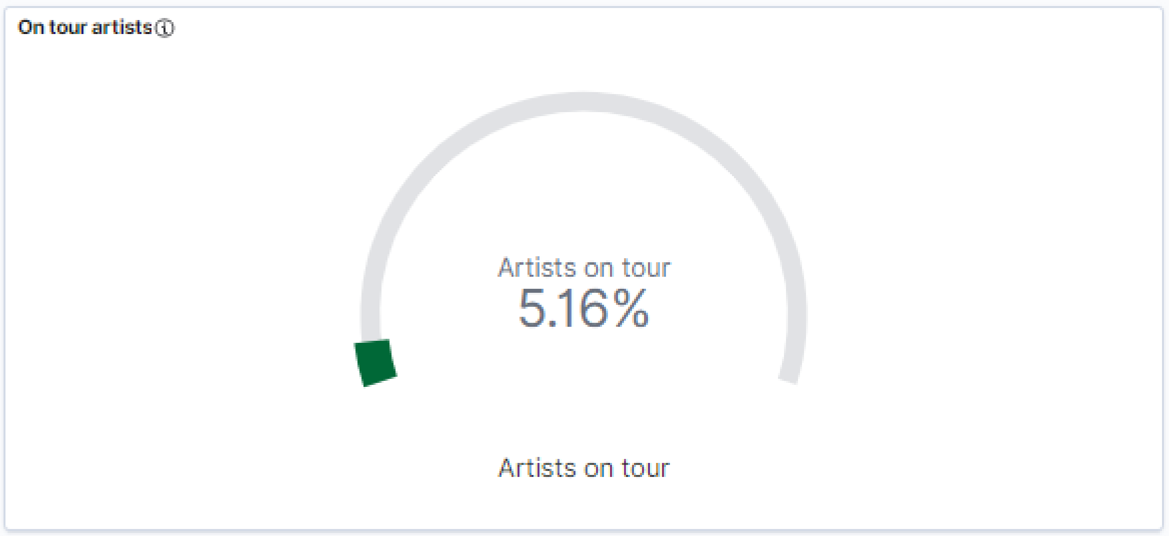
Number of listeners

123,006,878

Number of listeners

Most similar artists

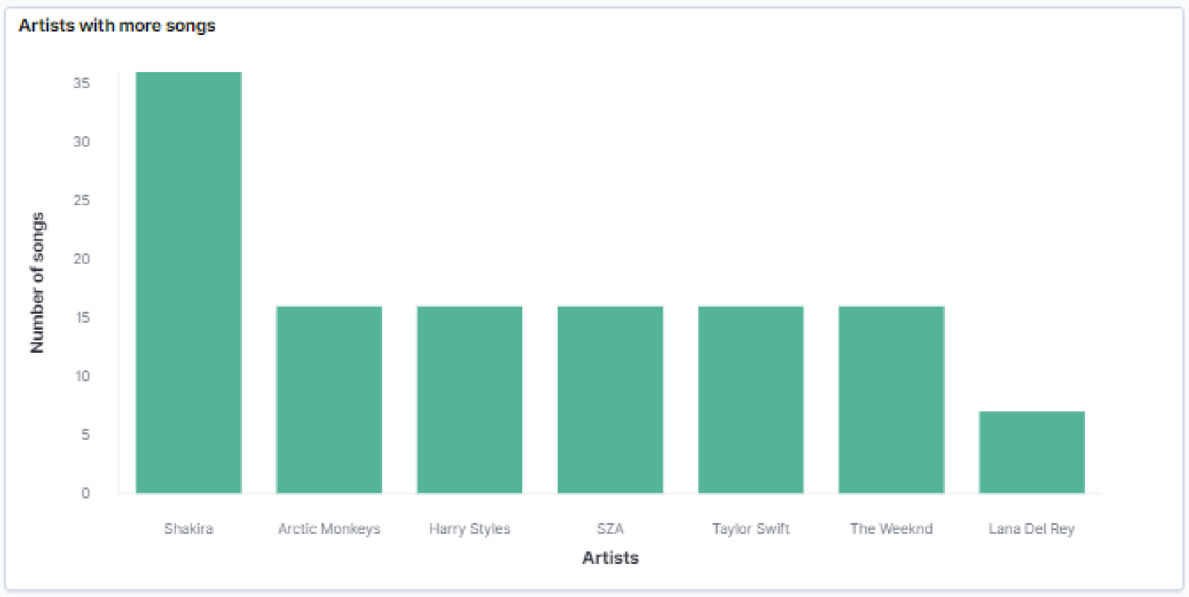
Belinda 20.000%	Jennifer Lopez 20.000%	Julieta Venegas 20.000%	Paulina Rubio 20.000%	Thalía 20.000%

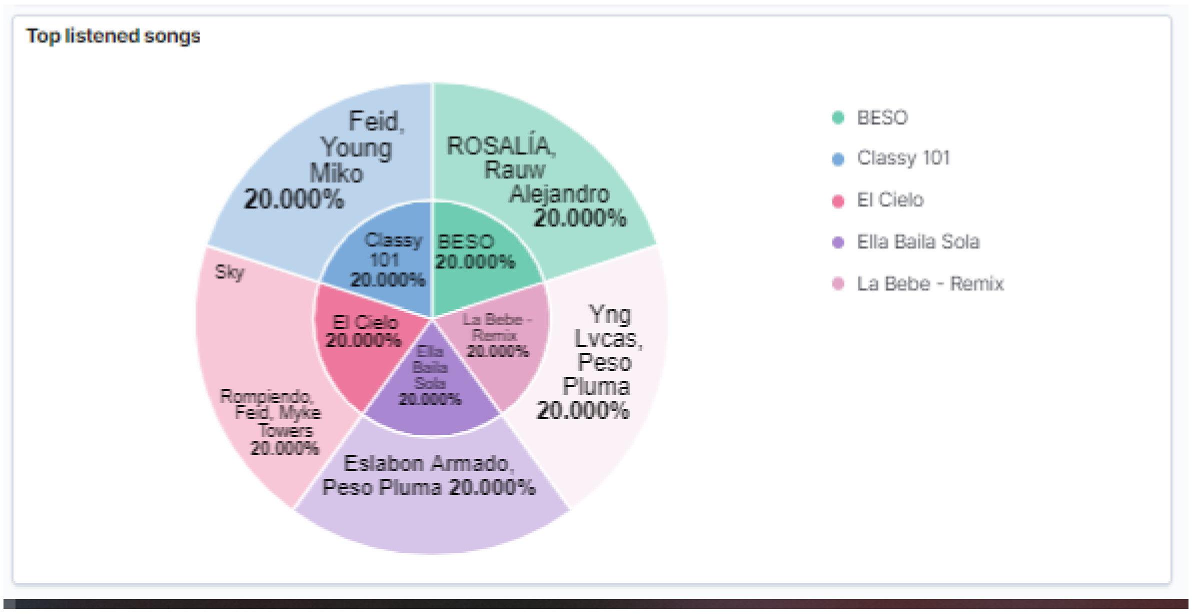


On tour artists

Artists on tour 5.16%

Artists on tour





4. Conclusion

In summary, this big data project successfully designed and implemented an end-to-end data architecture for music analysis. By using data from Last.fm and the Spotify API, the project has obtained valuable information about artists, top songs and music trends. A data pipeline orchestrated through Airflow efficiently ingested, formatted, combined, and indexed data for comprehensive analysis. The architecture of this project provided a solid foundation for processing and analyzing large amounts of music-related data. By indexing the combined data in Elasticsearch and visualizing it in Kibana, the results yielded valuable insights such as total listeners, touring artists, popular artists, and top songs. Overall, this project demonstrated the potential of big data and database technology in analyzing music-related data and uncovering meaningful patterns and trends in the industry.