INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Bases de Datos No Relacionales

PROYECTO 1

Aplicación con una base de datos NoSQL: Análisis de artistas en Spotify

Nombre del Grupo de Trabajo:

Top Tiers

Integrantes:

Maritrini Garcia Ruiz – 151490

Marco Antonio Chacon Amaro – 165681

Andrea Carolina Padilla Rodríguez – 166605

Fecha de entrega:

01 de octubre de 2020

Índice

Introducción

El mercado de la música en streaming crece día a día, aproximadamente siete de cada diez (67%) adultos han utilizado plataformas de streaming para escuchar música durante el último mes, según el estudio *Music Streaming Around the World*, de *Global Web Index* [1].

Dentro de las plataformas de música en streaming, Spotify es la más popular a nivel mundial; cuenta con más de 60 millones de canciones disponibles, 1.5 millones de podcasts y aproximadamente 299 millones de usuarios activos mensualmente [2].

Dada la alta penetración que tiene Spotify en el mercado de streaming musical, y a sus datos accesibles para desarrolladores, será la aplicación a utilizar en el presente proyecto.

Almacenar datos descargados de Spotify para su análisis se vuelve un gran reto, ya que cuenta con enormes volúmenes de datos disponibles. Utilizar las bases de datos relacionales tradicionales no es adecuado, debido a que se requiere una planificación cuidadosa y larga de su estructura, así como enfrentarse a datos no estructurados o nulos que vuelve esta opción altamente ineficiente.

El objetivo del presente proyecto será realizar un análisis para observar las tendencias actuales en la música de streaming, obteniendo la información de la aplicación Spotify, almacenándola de forma eficiente y generando gráficas para visualizar géneros y artistas, con sus respectivos valores de popularidad.

Solución

Para realizar el análisis planteado, se comenzó por investigar en diversas fuentes los URI de los artistas registrados en Spotify. Un URI (indicador uniforme de recursos) de Spotify es un enlace que se puede encontrar en el menú compartir de cualquier pista, álbum o perfil de artista en Spotify [3]. En Kaggle [4] existe una lista de 81,323 artistas únicos y sus URIs, como se muestra en la figura 1:

80110 unique values	81322 unique values
2:00 AM	spotify:artist:4tN3rZ7cChj4Wns2Wt2Nj6
2:15	spotify:artist:4HsOm6VNKZtGh8W8GhdNu4
2:54	spotify:artist:3LsQKoRgMc8VEkQn66jfAQ
4:20	spotify:artist:5KCG0FDMDPzQpxcohGUnyH
6:30	spotify:artist:5abblGojcZoe3zZBhJFBUi
6:33	spotify:artist:0oBPg2seHzVcAIOpdi1Ojj
10:32	spotify:artist:4CwN71HeOBgAcyk7nNKSTJ
4:00 PM	spotify:artist:6n4LC99DYefuU6icCPk1zU
2	spotify:artist:6pOCEbvtPkErkz3884ikYF

Fig. 1. Ejemplo de datos: nombres de artistas y sus respectivos URI.

Estos datos se descargaron en un archivo `.csv` que se importó a Python para su procesamiento.

Arquitectura del sistema

Para la solución, se utilizó la siguiente arquitectura:

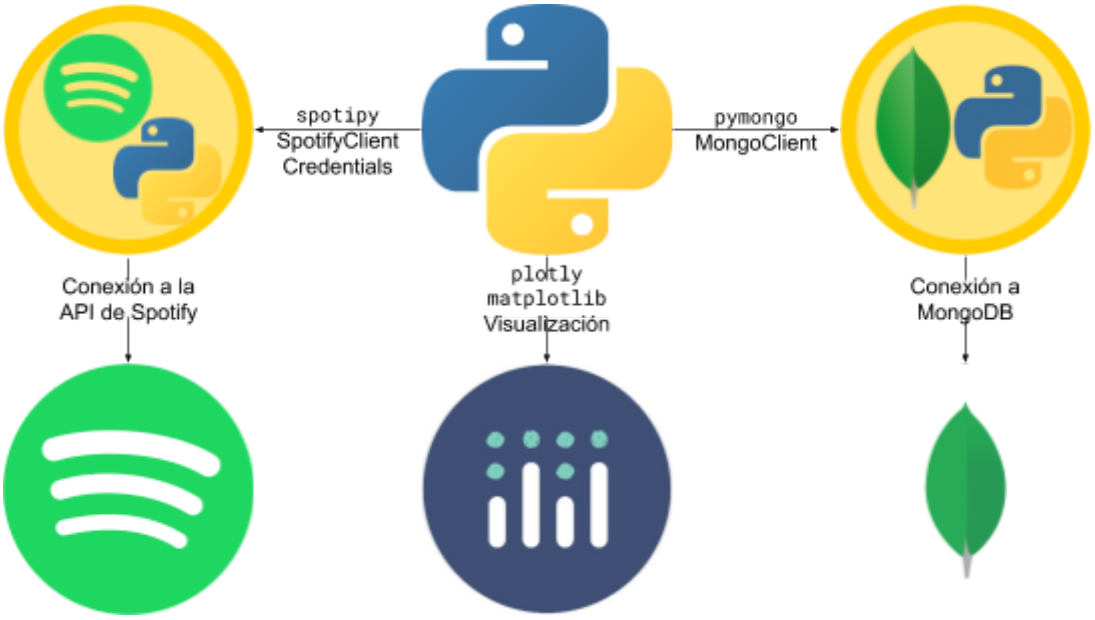

Fig. 2. Arquitectura del sistema.

En Python, por medio de `spotipy` [5], se realizó la conexión a la API de Spotify, incluyendo las credenciales necesarias para acceder a la API, las cuales se pueden obtener en Spotify Developers [6]. Después, se realizó la búsqueda de cada URI

para obtener la información del artista. Luego, se hizo uso de `pymongo` [7] para establecer la conexión con MongoDB y manejar la base de datos NoSQL creada con los datos obtenidos en la búsqueda. Para mostrar los datos de manera gráfica, se utilizaron `plotly` [8] y `matplotlib` [9].

Características de la solución

Funcionalidades principales

La funcionalidad de la solución puede dividirse en cinco segmentos:

1. Conexión con la API de Spotify a través de las credenciales Client ID y Client Secret.

Fig. 3. Credenciales obtenidas a través de Spotify Developers

2. Obtención de información en formato JSON para cada artista (URI) contenido en el dataset [4].


```
{'external_urls': {'spotify': 'https://open.spotify.com/artist/1r4hJ1h58CWwUQe3MxPuau'},
 'followers': {'href': None, 'total': 20894502},
 'genres': ['reggaeton', 'reggaeton colombiano'],
 'href': 'https://api.spotify.com/v1/artists/1r4hJ1h58CWwUQe3MxPuau',
 'id': '1r4hJ1h58CWwUQe3MxPuau',
 'images': [{'height': 640,
   'url': 'https://i.scdn.co/image/0b6f5467bbde78796b33d8eed85157620b3080cc',
   'width': 640},
  {'height': 320,
   'url': 'https://i.scdn.co/image/6265d3a759f85b133744209341c9bc87f75d919c',
   'width': 320},
  {'height': 160,
   'url': 'https://i.scdn.co/image/bb1ae742982c6e4d90c8d40d1f860b1e3f89e7d2',
   'width': 160}],
 'name': 'Maluma',
 'popularity': 95,
 'type': 'artist',
 'uri': 'spotify:artist:1r4hJ1h58CWwUQe3MxPuau'}
```


Fig. 4. Ejemplo de un dato JSON generado a partir del URI de cada artista.

3. Creación de la base de datos en MongoDB, insertando cada objeto JSON obtenido.

4. Elaboración de consultas a la base de datos de MongoDB para conocer:
    a. La cantidad de registros en la colección `artistas`.
    b. La cantidad de artistas con algunos de los géneros más comunes: `pop, rock, classical, rap y hip hop.`
    c. La cantidad de artistas con menos de cierta cantidad de puntos de popularidad.

5. Elaboración de gráficos para visualizar:
    a. La distribución estadística de la popularidad de los artistas.
    b. La distribución poblacional de la cantidad de seguidores de los artistas.
    c. Los géneros más populares de los artistas más populares.
    d. Los géneros más populares de los artistas menos populares.
    e. Las imágenes de los artistas más populares.

Obtención y proceso de instalación de las herramientas nuevas utilizadas

Para poder implementar la solución y extraer los datos de la API de Spotify, fue necesario crear una cuenta y consultar las credenciales Client ID y Client Secret (figura 3) para poder autenticarnos como clientes, establecer la conexión y generar peticiones.

A través de la interfaz de Anaconda, se instalaron nuevas librerías necesarias para el proyecto ; a continuación se listan dichas librerías con una breve descripción:

- **`spotipy:`** librería de Python para la API web de Spotify. Gracias a esta librería se obtiene acceso completo a todos los datos musicales proporcionados por la plataforma Spotify [5]. Se puede instalar desde la terminal mediante el comando `pip install spotipy`, o en Anaconda con `conda install -c jkroes spotipy.`

- **`pandas:`** herramienta de manipulación de datos de alto nivel desarrollada por Wes McKinney [10]. Su estructura de datos clave se llama DataFrame, que

permiten almacenar y manipular datos tabulares en filas de observaciones y columnas de variables. Se puede instalar desde la terminal mediante el comando `pip install pandas`, o en Anaconda con `conda install -c anaconda pandas`.

- **`json`:** librería de Python que permite la codificación y decodificación de JSON. El método `dump()` de esta librería nos permitió convertir objetos de Python a JSON. Disponible en las librerías estándar de Python [11].

- **`plotly`:** librería de Python capaz de crear gráficas interactivas de alta calidad [8]. Se puede instalar desde la terminal mediante el comando `pip install plotly==4.10.0`, o en Anaconda con `conda install -c plotly plotly=4.10.0`. Para usar en Jupyter Notebook, se deben instalar los paquetes `notebook` y `ipywidgets` mediante el comando `pip install "notebook>=5.3" "ipywidgets==7.5"`, o en Anaconda con `conda install "notebook>=5.3" "ipywidgets=7.5"`.

- **`matplotlib`:** librería de Python para la generación de gráficos a partir de datos contenidos en listas o arrays [9]. Se puede instalar desde la terminal mediante el comando `pip install matplotlib`, o en Anaconda con `conda install -c conda-forge matplotlib`.

- **`collections`:** librería de Python que implementa tipos de datos de contenedores especializados que brindan alternativas a los contenedores integrados de uso general de Python, `dict`, `list`, `set` y `tuple` [12]. De aquí se importa el objeto `Counter()`: una colección desordenada donde los elementos se almacenan como claves de diccionario y sus recuentos se almacenan como valores de diccionario. Se utiliza en la solución para contar los elementos de una lista. Disponible en las librerías estándar de Python.

Obtención y almacenamiento de los datos

Obtención y proceso de transformación de los datos

Se leyó en Python el archivo `.csv` y se guardó su información en la variable `artists_uris`, tal cual muestra la siguiente figura:

```
In [7]: artists_uris = pd.read_csv("artists_uris.csv", names=['name','uri'])
        print(artists_uris.shape)
        artists_uris.head()

        (81323, 2)

Out[7]:
```

	name	uri
0	1:43	spotify:artist:39EHxSQAIaWusRqSI9xoyF
1	2:00 AM	spotify:artist:4tN3rZ7cChj4Wns2Wt2Nj6
2	2:15	spotify:artist:4HsOm6VNKZtGh8W8GhdNu4
3	2:54	spotify:artist:3LsQKoRgMc8VEkQn66jfAQ
4	4:20	spotify:artist:5KCG0FDMDPzQpxcohGUnyH

Fig. 5. Lectura de los nombres de los artistas y sus URIs de Spotify.

Al consultar desde Python la información de cada artista en Spotify por medio de su URI, con la función `artists(artists)` de `spotipy` [13], se obtiene la información mostrada en la figura 4. Con estos datos, se creó el archivo tipo JSON de tamaño `84.4 MB` llamado `data_artists.json`, que contiene la información de los 81,323 artistas consultados. En la solución aparece de la siguiente manera:

```python
import json

# función para escribir en el archivo JSON
def write_json(data, filename='data_artists.json'):
    with open(filename,'w') as f:
        json.dump(data, f, indent=4)

# consultamos y almacenamos la información por artista
with open('data_artists.json') as json_file:
    artists_uris_list = list(artists_uris['uri'])
    data = []

    for i in range(0,len(artists_uris_list),50):
        x = sp.artists(artists_uris_list[i:i + 50])['artists']
        data = data + x
# Escribimos los datos en un archivo JSON
write_json(data)
```

Fig. 6. Búsqueda de la información de cada artista, y creación y llenado del archivo JSON.

Almacenamiento de los datos

Teniendo los datos en formato JSON, se pobló al manejador para poder realizar las consultas:

```
from pymongo import MongoClient as Connection

connection = Connection('localhost',27017)
mydb = connection["spotify_data"]
mycol = mydb["artistas"]
mycol.drop() # borramos la colección por si ya existe
mycol = mydb["artistas"]

with open('data_artists.json') as json_file:
    artist_data = json.load(json_file)

# insertamos datos
mycol.insert_many(artist_data)
```

Fig. 7. Creación de la base "`spotify_data`" y su colección "`artistas`", en donde se insertaron los datos.

Como indica la figura anterior, la información se almacenó en la base de datos `spotify_data`, en la colección `artistas`, visualizandose de la siguiente manera:

Fig. 8. Ejemplo de datos almacenados en MongoDB: información obtenida de *The Beatles* a través de Spotify.

Estructura de las “tablas” de la BD

Los datos recopilados de cada artista, e insertados en la colección, son los que forman el `artist object` de Spotify. La llave, el tipo y la descripción de cada valor que conforma dicho objeto se muestran en la siguiente tabla:

Tabla 1
Definición del `artist object` de Spotify

Llave	Tipo de valor	Descripción
`external_urls`	Un `external urls object`	URL externas conocidas de este artista.
`followers`	Un `followers object`	Información sobre los seguidores del artista.
`genres`	Arreglo de `strings`	Una lista de los géneros con los que está asociado el artista. Por ejemplo: `"Prog Rock"`, `"Post-Grunge"`. (Si aún no está clasificado, el arreglo está vacío).
`href`	`string`	Un link al Web API endpoint que proporciona todos los detalles del artista.
`id`	`string`	El ID del artista en Spotify.
`images`	Arreglo de `image object`	Imágenes del artista en varios tamaños, empezando por la más amplia.
`name`	`string`	El nombre del artista
`popularity`	`int`	La popularidad del artista. El valor estará entre 0 y 100, siendo 100 el más popular. La popularidad del artista se calcula a partir de la popularidad de todas las pistas del artista.
`type`	`string`	El tipo de objeto: `"artist"`
`uri`	`string`	El URI de Spotify para el artista.

Estos valores son los que se convierten a formato JSON para ingresarlos a MongoDB y realizar las consultas.

Resultados

Consultas

Se realizaron cinco consultas diferentes utilizando la colección `artistas` en MongoDB:

1. Se obtuvo la cantidad de documentos en la colección `artistas`, obteniendo como resultado 81,323 artistas en total:

```
# Cantidad total de documentos en la colección de artistas
cantidad_documentos = mycol.count_documents({})
print(f"La cantidad de documentos en la colección de artistas es de: {cantidad_documentos}")
```

```
La cantidad de documentos en la colección de artistas es de: 81323
```

Fig. 9. Código de python para consultar la cantidad de documentos en la colección de artistas.

2. Se obtuvo la cantidad de artistas con los géneros pop, rock, classical, rap y hip hop:

```
genres = ['pop', 'rock', 'classical', 'rap', 'hip hop']

# Cantidad total de documentos que contienen diversos géneros
for genre in genres:
    cantidad = mycol.count_documents({ "genres": genre})
    print(f"La cantidad de artistas con el género {genre} es de: {cantidad}")
```

```
La cantidad de artistas con el género pop es de: 557
La cantidad de artistas con el género rock es de: 581
La cantidad de artistas con el género classical es de: 210
La cantidad de artistas con el género rap es de: 464
La cantidad de artistas con el género hip hop es de: 478
```

Fig. 10. Código de python para consultar la cantidad de artistas de diferentes géneros.

3. Se obtuvo la cantidad de artistas con menos de ciertos puntos de popularidad, definidos en una variable aparte (`popularity_threshold`). En el ejemplo siguiente, el parámetro fue de 30 puntos de popularidad, resultando en 36,346 artistas:

```
# Cantidad de artistas con popularidad menor a un cierto un cierto umbral
popularity_threshold = 30
cantidad = mycol.count_documents({ "popularity": { "$lt": popularity_threshold } })
print(f"La cantidad de artistas con menos de {popularity_threshold} puntos de popularidad es de: {cantidad}")
```

```
La cantidad de artistas con menos de 30 puntos de popularidad es de: 36346
```

Fig. 11. Código de python para consultar la cantidad de artistas con menos de 30 puntos de popularidad.

4. Se obtuvieron los géneros que cantan los artistas más populares. El resultado de esta consulta se muestra en la figura 16.

```python
from collections import Counter

# consultamos los generos de los artistas que rebasan un cierto umbral de popularidad
popularity_threshold = 80
lista_generos = mycol.find({ "popularity": { "$gt": popularity_threshold } }, {"genres":1, "_id":0})

# liampiamos un poco los datos
lista_generos = [e for genero in lista_generos for e in genero['genres']]
diccionario_generos = Counter(lista_generos)
```

```python
import plotly.express as px

genres_threshold = 20

nombre_genero = [e[0] for e in diccionario_generos.most_common()][:genres_threshold]
valor_genero = [e[1] for e in diccionario_generos.most_common()][:genres_threshold]

fig = px.bar(x=nombre_genero,
             y=valor_genero,
             labels = {"x":"género","y":"cantidad"},
             title="Géneros más comunes en los artistas más populares",
             color=valor_genero,
             opacity=0.7
            )
fig.show()
```

Fig. 12. Código de python para consultar y graficar los géneros que cantan los artistas más populares.

5. Se obtuvieron los géneros que cantan los artistas menos populares. El resultado de esta consulta se muestra en la figura 17.

```python
# consultamos los generos de los artistas que no rebasan un cierto umbral de popularidad
popularity_threshold = 10
lista_generos = mycol.find({ "popularity": { "$lt": popularity_threshold } }, {"genres":1, "_id":0})

# limpiamos los datos
lista_generos = [e for genero in lista_generos for e in genero['genres']]
diccionario_generos = Counter(lista_generos)

# graficamos
genres_threshold = 10

nombre_genero = [e[0] for e in diccionario_generos.most_common()][:genres_threshold]
valor_genero = [e[1] for e in diccionario_generos.most_common()][:genres_threshold]

fig = px.bar(x=nombre_genero,
             y=valor_genero,
             labels = {"x":"género","y":"cantidad"},
             title="Géneros más comunes en los artistas menos populares",
             color=valor_genero,
             opacity=0.7
            )
fig.show()
```

Fig. 13. Código de python para consultar y graficar los géneros que cantan los artistas menos populares.

Gráficos

También se generaron cuatro gráficas dinámicas y una planilla de imágenes con base en la colección `artistas` en MongoDB.

1. Distribución estadística de la popularidad de los artistas.

La siguiente gráfica muestra la distribución estadística de la popularidad de los artistas, valor comprendido entre 0 y 100. Se utilizó la librería `plotly` para que dinámicamente se ajuste el rango de la gráfica con ayuda de la barra localizada en la parte inferior de la gráfica.

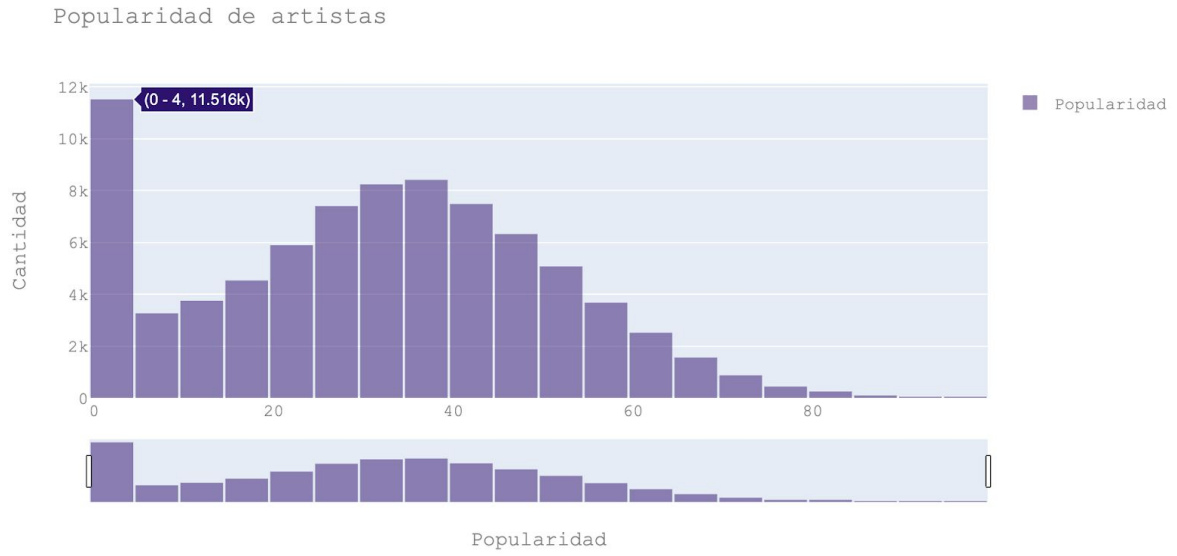

Fig. 14. Distribución estadística de la popularidad de los artistas.

Es posible descubrir, al pasar el cursor sobre la gráfica, que la moda de popularidad en los artistas es cero, ya que, de los 81,323 artistas, 11,516 cumplen con esta característica, siendo el 14 % de los artistas.

2. Diagrama de caja y brazos para la distribución poblacional de la cantidad de seguidores de los artistas.

La figura 15 muestra un diagrama de caja y brazos para la distribución poblacional de la cantidad de seguidores de los artistas. Colocando el cursor sobre la gráfica, se pueden observar distintos parámetros:

a. Mínimo: 9.11 millones.
b. Primer cuartil (acumula el 25% de la distribución): 11.3 millones.
c. Mediana (acumula el 50% de la distribución): 15.15 millones.
d. Tercer cuartil (acumula el 75% de la distribución): 22.38 millones.
e. Máximo: 70.01 millones.

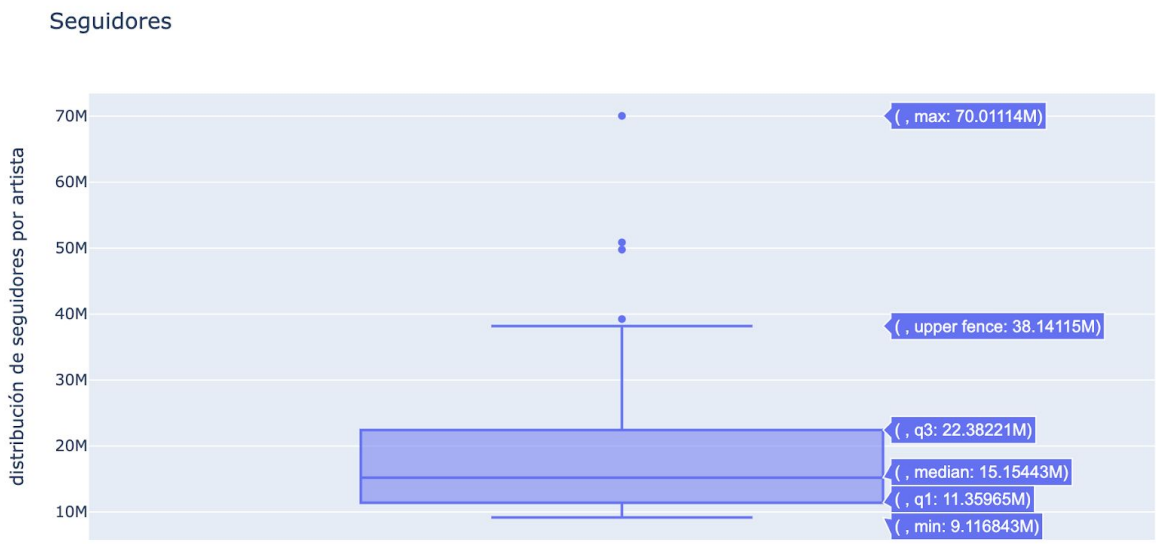

Fig. 15. Diagrama de caja y brazos para la distribución poblacional de la cantidad de seguidores de los artistas

3. Géneros más comunes entre los artistas más populares.

La figura 16 ilustra los veinte géneros más comunes entre los artistas más populares. Se definió primero un umbral de popularidad para los artistas, que en este caso fue 80, posteriormente, se generó una consulta que devuelve únicamente el nombre de los géneros si la popularidad del artista es mayor al valor. Como se muestra en la gráfica, el género más popular es el pop, pues, de los artistas con popularidad superior a 80 puntos, 128 pertenecen a esta categoría.

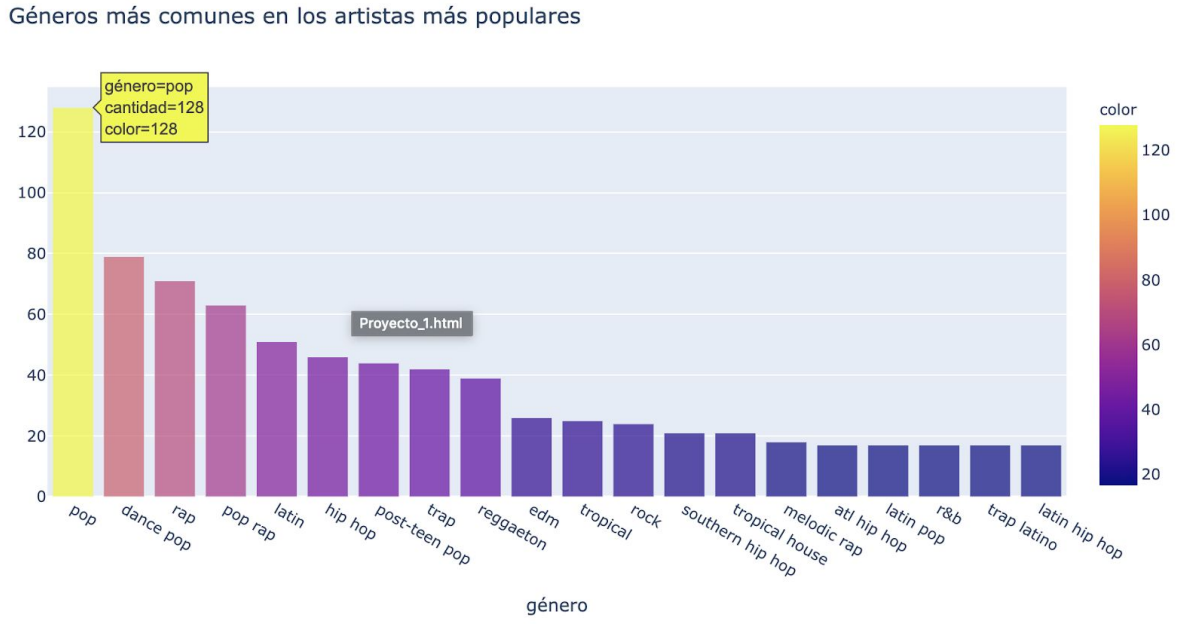

Fig. 16. Géneros más comunes entre los artistas más populares

4. Géneros más comunes entre los artistas menos populares.

La figura 17 es muy similar a la gráfica anterior, sin embargo, ahora ilustra los 10 géneros más comunes entre los artistas menos populares. Se definió primero un umbral de popularidad para los artistas, que en este caso fue 10, posteriormente, se generó una consulta que devuelve únicamente el nombre de los géneros si la popularidad del artista es mayor al valor. Como se muestra en la gráfica, el género más popular es el *string quartet*, pues, de los artistas con popularidad menor a 10 puntos, 65 pertenecen a esta categoría.

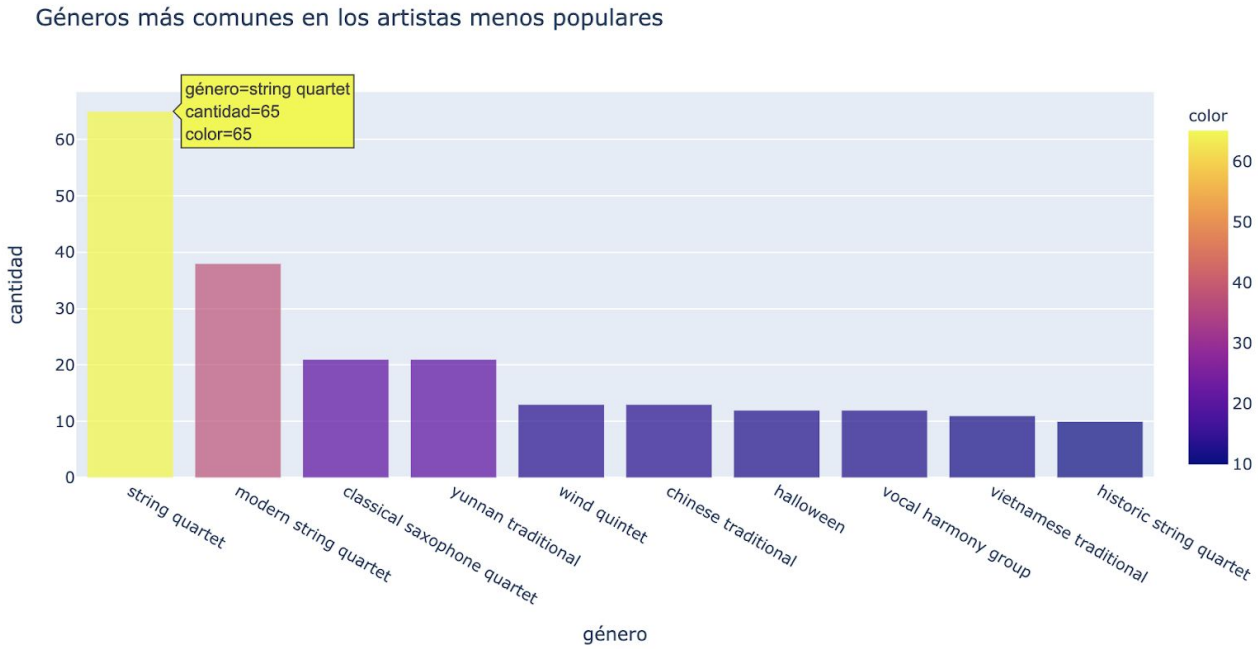

Fig. 17. Géneros más comunes entre los artistas menos populares

5. Planilla de imágenes de los artistas más populares.

Por último, se creó una planilla con las imágenes de los artistas más populares, junto con su ranking de popularidad. Se estableció un umbral de popularidad para después elaborar una consulta, asegurando que la popularidad de los artistas estuviera por encima de dicho umbral. Obtener la imagen para los artistas fue posible gracias al atributo `images` en la colección `artistas`, pues dentro de ese arreglo está alojado el URL de cada imagen, que pudo ser recuperada a través del método `requests.get(URL)`.

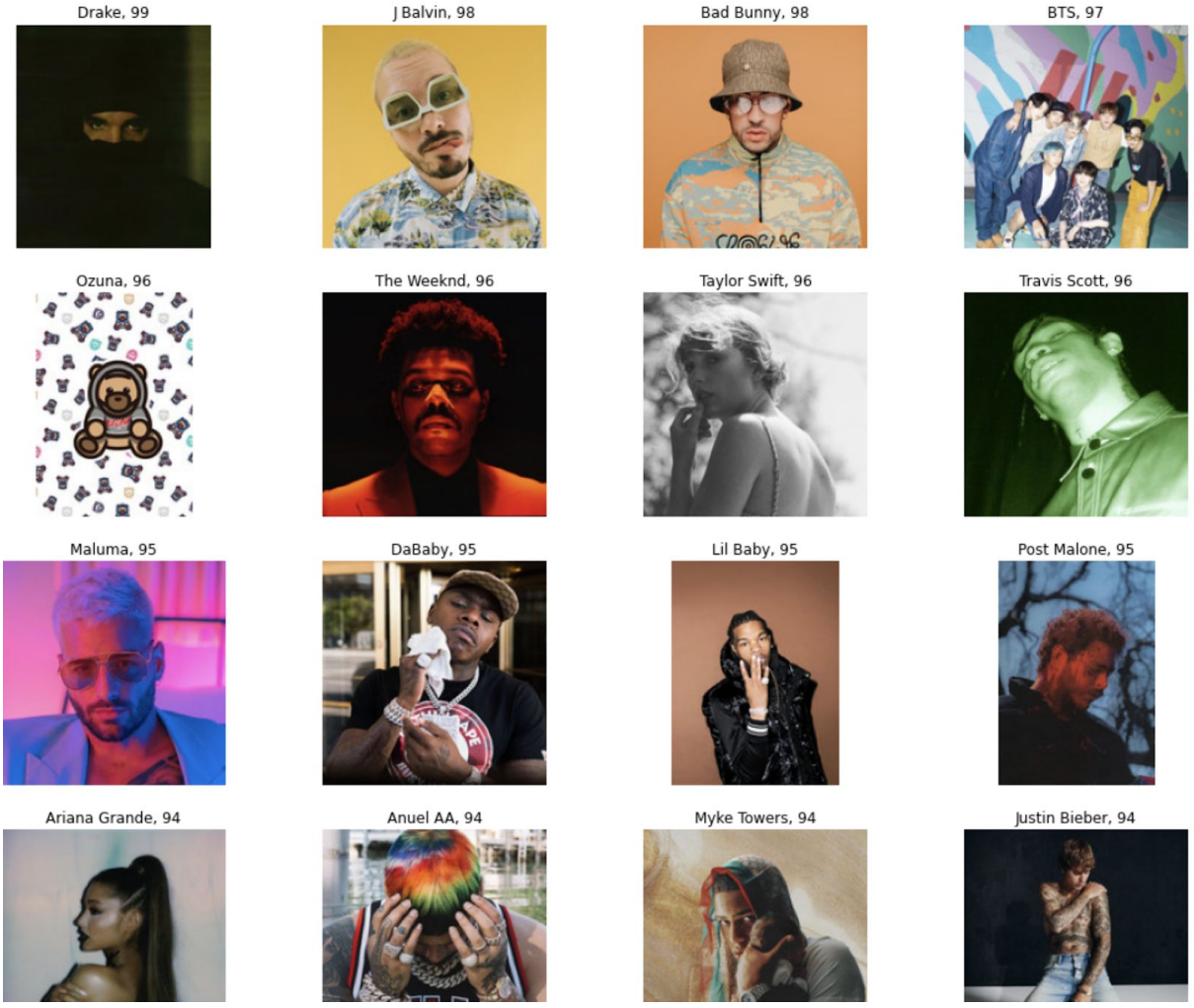

Fig. 18. Dashboard de los artistas más populares

Conclusiones

A través de las diferentes consultas y gráficos elaborados, fue posible detectar las tendencias actuales en géneros y artistas en la música de streaming, siendo el pop uno de los géneros más escuchados actualmente.

Además, se pusieron a prueba los beneficios de utilizar bases de datos no relacionales en contraste con las relacionales, pues ahorramos una cantidad considerable de tiempo al no tener que definir múltiples tablas y relaciones. Otro aspecto positivo fue la velocidad en las consultas, ya que acceder a cada elemento resultó rápido gracias a que se construyó la colección de acuerdo a la estructura obtenida de Spotify. Además, se hizo uso de nuevas herramientas que, en el futuro, serán de gran utilidad.

Resultó muy satisfactorio hacer uso de todos los conocimientos vistos hasta ahora en la materia, pues tuvimos la oportunidad de ponerlos en práctica y reflexionar sobre la importancia y la utilidad que tienen las bases de datos NoSQL al trabajar con una gran cantidad de información usando datos no estructurados.

Bibliografía

[1] Global Web Index, "Infographic: Music Streaming Around the World", *Global Web Index*, 2020. [Online]. Available: https://www.globalwebindex.com/reports/music-streaming-around-the-world. [Accessed: 30- Sep- 2020].

[2] Spotify AB, "Company Info", *Spotify*, 2020. [Online]. Available: https://newsroom.spotify.com/company-info/#:~:text=Today%2C%20Spotify%20is%20the%20world's,138m%20subscribers%2C%20across%2092%20markets. [Accessed: 30- Sep- 2020].

[3] Spotify AB, "What's a Spotify URI?", *Spotify Community*, 2018. [Online]. Available: https://community.spotify.com/t5/Spotify-Answers/What-s-a-Spotify-URI/ta-p/919201. [Accessed: 30- Sep- 2020].

[4] E. Call, "Spotify Artists", *Kaggle*, 2017. [Online]. Available: https://www.kaggle.com/ehcall/spotify-artists. [Accessed: 30- Sep 2020].

[5] P. Lamere, "Welcome to Spotipy! — spotipy 2.0 documentation", *Spotipy*, 2014. [Online]. Available: https://spotipy.readthedocs.io/en/2.16.0/. [Accessed: 30- Sep- 2020].

[6] Spotify AB, "Platform Documentation", *Spotify for Developers*, 2020. [Online]. Available: https://developer.spotify.com/documentation/. [Accessed: 30- Sep- 2020].

[7] MongoDB, Inc., "PyMongo 3.11.0 Documentation", *Pymongo*, 2020. [Online]. Available: https://pymongo.readthedocs.io/en/stable/. [Accessed: 30- Sep- 2020].

[8] Plotly Technologies Inc., "Plotly: The front-end for ML and data science models", *Plotly*, 2020. [Online]. Available: https://plotly.com/. [Accessed: 30- Sep- 2020].

[9] The Matplotlib development team, "Matplotlib 3.3.2 documentation", *Matplotlib*, 2020. [Online]. Available: https://matplotlib.org/. [Accessed: 30- Sep- 2020].

[10] learnpython.org, "Pandas Basics", *Learn Python - Free Interactive Python Tutorial*. [Online]. Available: https://www.learnpython.org/en/Pandas_Basics#:~:text=Pandas%20is%20a%20high%2Dlevel,observations%20and%20columns%20of%20variables. [Accessed: 30- Sep- 2020].

[11] Python Software Foundation, "JSON encoder and decoder", *Python 3.8.6 documentation*, 2020. [Online]. Available: https://docs.python.org/3/library/json.html. [Accessed: 30- Sep- 2020].

[12] Python Software Foundation, "8.3. collections — Python 2.7.18 documentation", *The Python Standard Library*, 2020. [Online]. Available: https://docs.python.org/2/library/collections.html. [Accessed: 30- Sep- 2020].

[13] Spotify AB, "Get an Artist", *Spotify for Developers*, 2020. [Online]. Available: https://developer.spotify.com/documentation/web-api/reference/artists/get-artist/. [Accessed: 30- Sep- 2020].