

Pontificia Universidad Javeriana

Semestre 2310

Gestión De Datos

Integrantes:

- Jorge Esteban Castañeda López
- Miguel Ángel Gutiérrez Ibagué
- Abelardo Valdivieso Acevedo

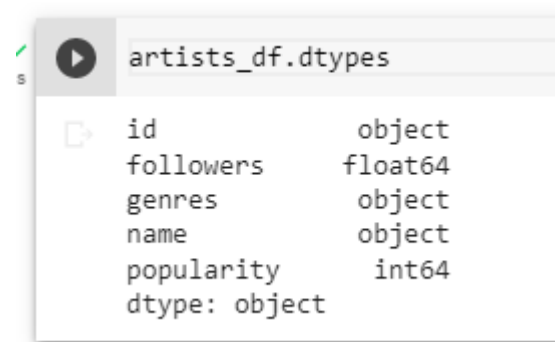
Github: <https://github.com/miguelgutierrez/proyecto-gestion-datos-2310>

1. Descripción de los datos

Se cuenta con dos datasets principales, uno asociado a los artistas en Spotify, y otro asociado a las canciones. La descripción de los mismos, se realiza a partir de Pandas como un perfilamiento de datos, y, de una descripción de los esquemas presentados.

Artistas

El dataset de artistas cuenta con los siguientes tipos de datos:



```
artists_df.dtypes
id                object
followers         float64
genres            object
name              object
popularity        int64
dtype: object
```

De manera general estos campos pueden ser descritos de la siguiente manera:

- **id:** Identificador único del artista en Spotify. *Tipo de variable:* Cualitativa nominal.
- **followers:** Cantidad de seguidores del artista. *Tipo de variable:* Cuantitativa discreta.
- **genres:** Lista con los géneros asociados a las canciones del artista. *Tipo de variable:* Cualitativa nominal.
- **name:** Nombre del artista. *Tipo de variable:* Cualitativa nominal.
- **popularity:** Valor numérico que representa la popularidad del artista. *Tipo de variable:* Cuantitativa discreta.

Canciones

El dataset de canciones cuenta con los siguientes tipos de datos:

```
tracks_df.dtypes
```

```
id            object
name          object
popularity    float64
duration_ms   int64
explicit      int64
artists       object
id_artists    object
release_date  object
danceability  float64
energy        float64
key           float64
loudness      float64
mode          float64
speechiness   float64
acousticness  float64
instrumentalness float64
liveness      float64
valence       float64
tempo         float64
time_signature float64
dtype: object
```

De manera general estos campos pueden ser descritos de la siguiente manera:

- **id:** Id de la canción en Spotify. *Tipo de variable:* Cualitativa nominal.
- **name:** Nombre de la canción. *Tipo de variable:* Cualitativa nominal.
- **popularity:** Valor numérico que representa la popularidad de la canción. *Tipo de variable:* Cuantitativa discreta.
- **duration_ms:** Longitud de la canción en milisegundos. *Tipo de variable:* Cualitativa discreta
- **explicit:** Valor numérico (booleano) entre 0 y 1 que representa si la canción tiene letra explícita o no. *Tipo de variable:* Cualitativa nominal
- **artists:** Arreglo con el nombre de los artistas asociados a la canción. *Tipo de variable:* Cualitativa nominal
- **id_artists:** Arreglo con los ids de los artistas asociados a la canción. *Tipo de variable:* Cualitativa nominal
- **release_date:** Fecha en la que se publicó la canción. *Tipo de variable:* Cualitativa ordinal
- **danceability:** Valor flotante entre 0 y 1 que determina que tan “bailable” es una canción. *Tipo de variable:* Cuantitativa ordinal
- **energy:** Valor flotante entre 0 y 1 que determina la intensidad de una canción. *Tipo de variable:* Cuantitativa ordinal
- **key:** Valor que identifica la clave musical de la canción. *Tipo de variable:* Cuantitativa nominal

- **loudness:** Nivel de la música representado en decibeles, típicamente entre -60db y 0db. *Tipo de variable:* Cuantitativa ordinal
- **mode:** Indica la modalidad (mayor o menor) de una canción. Valor entre 0 y 1. *Tipo de variable:* Cuantitativa ordinal
- **speechiness:** Valor que permite identificar que tan “hablada” es una canción, de manera que es posible detectar si puede ser un podcast o una canción. *Tipo de variable:* Cuantitativa ordinal
- **acousticness:** Valor flotante entre 0 y 1 que determina que tan acústica de una canción. *Tipo de variable:* Cuantitativa ordinal
- **instrumentalness:** Valor flotante entre 0 y 1 que permite identificar si una canción es instrumental o cantada. *Tipo de variable:* Cuantitativa ordinal
- **liveness:** Valor flotante entre 0 y 1 que permite predecir si una canción fue grabada o no en vivo. *Tipo de variable:* Cuantitativa ordinal
- **valence:** Valor flotante entre 0 y 1 que permite identificar que tan “feliz” es una canción. *Tipo de variable:* Cuantitativa ordinal
- **tempo:** Valor numérico que representa los BPM de una canción. *Tipo de variable:* Cuantitativa discreta
- **time_signature:** Valor que representa la firma de tiempo de una canción ($\frac{3}{4}$, $\frac{6}{8}$, etc). *Tipo de variable:* Cualitativa discreta

2. Análisis de calidad de datos y proceso de limpieza

Los resultados de la limpieza de datos se pueden evidenciar más claramente en el notebook adjunto en el repositorio de Github adjunto en las referencias del presente documento. Particularmente en la carpeta Entregas 1 y 2, notebook 1. Sin embargo, de manera general se comparten los pasos ejecutados:

Artistas

- Validar la existencia de nulos en las columnas.
- Reemplazar los nulos en followers con -1.
- Validar posibles datos con formatos variables. En este caso géneros y popularidad.

Tracks

- Validar la existencia de nulos en las columnas.
- Reemplazar los nulos con valores relevantes según sea el caso.
- Validar todos los formatos numéricos, en caso de que alguno no coincida
- Validar release_date como formato correcto
- Al tener data inválida, se crea función de detección de errores en el formato y se obtienen los valores inválidos
- Crear función para reemplazar los múltiples casos de fechas erróneas encontradas en el dataset.
- Aplicar la función a la columna de release_date.
- Volver a revisar si hay errores

3. Análisis exploratorio de datos

Los resultados del análisis tanto univariado como bivariado se pueden evidenciar en el notebook adjunto en el repositorio de Github adjunto en las referencias del

presente documento. Particularmente en la carpeta Entregas 1 y 2, notebook 1. Sin embargo, se presentan las métricas usadas para cada uno de los tipos de análisis realizados:

Análisis univariado

Artistas

- En un análisis inicial, se evidencia que la media de followers de los artistas es de 10220. Sin embargo esto se debe al gran número de artistas que cuentan con 0 followers, tal como se puede evidenciar en el histograma presentado en el notebook.
- El artista con más followers es Ed Sheeran con un total de 78900234. Lo que concuerda con el histograma en donde se podría considerar incluso como un valor atípico.
- La media de popularidad es de 8.79, siendo esto concordante con el histograma obtenido, de manera que, a más popularidad, menor cantidad de artistas. Coincidiendo con Justin Bieber como artista con mayor popularidad, siendo esta de 100.

Tracks

- Se obtienen resultados tales como, la canción más popular es Peaches. En promedio, una canción dura 3,8 minutos. El cantante con mayor cantidad de canciones es Die Drei.
- En la mayor cantidad de métricas, se cuenta con un valor alto de valores que eran inicialmente nulos.
- Existen más podcasts que canciones según el análisis inicial.

Análisis bivariado

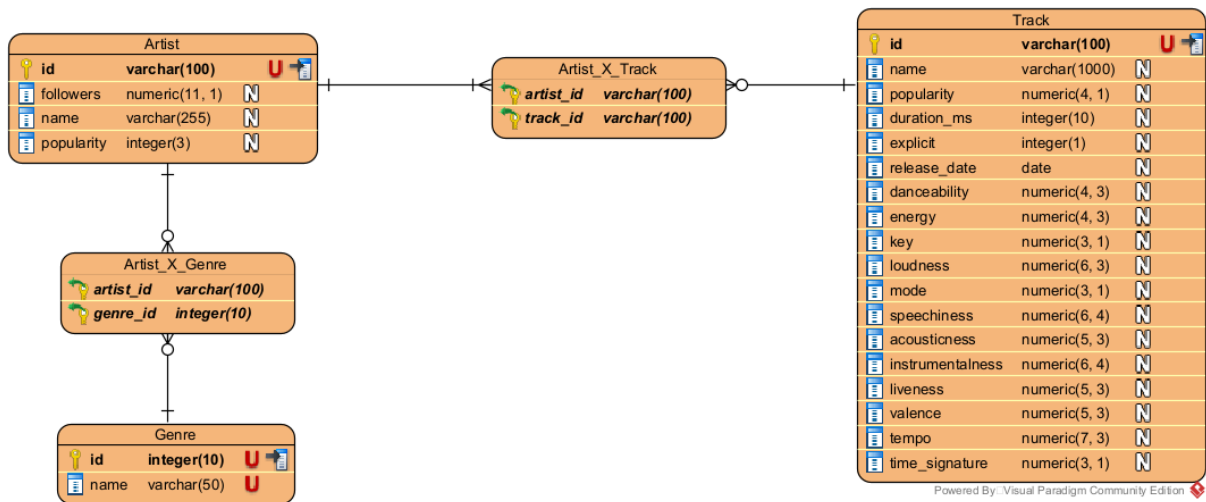
Artistas

- Se realiza un análisis bivariado sobre followers vs popularidad, encontrando que existe una correlación monótona más no lineal. Entendiendo que, pese a que puede existir una relación entre el aumento de seguidores y popularidad, esto no es constante.

Tracks

- La correlación más alta entre las diferentes métricas de tracks, se encuentra entre loudness y energía, siendo esta una relación tanto monótona como lineal.
- En su mayoría la correlación entre las diferentes dimensiones de un track, son cercanas a 0. Por lo tanto, es difícil predecir los comportamientos de dichas dimensiones a partir de otras dimensiones.

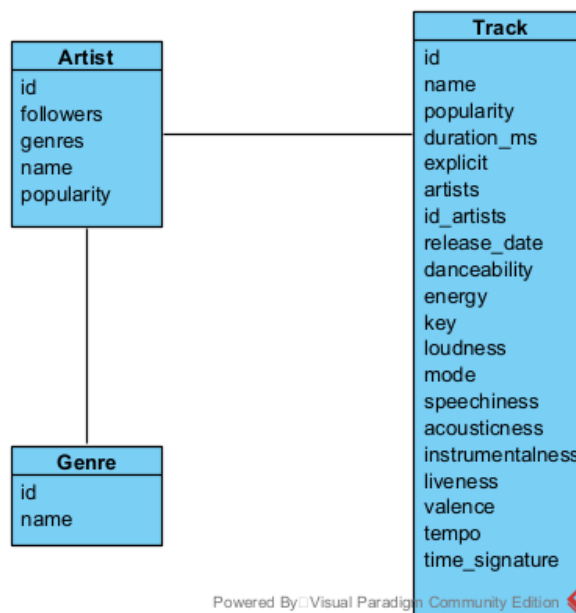
4. Modelo de datos relacional



Para el modelo relacional se cuenta con 5 tablas, enunciadas a continuación:

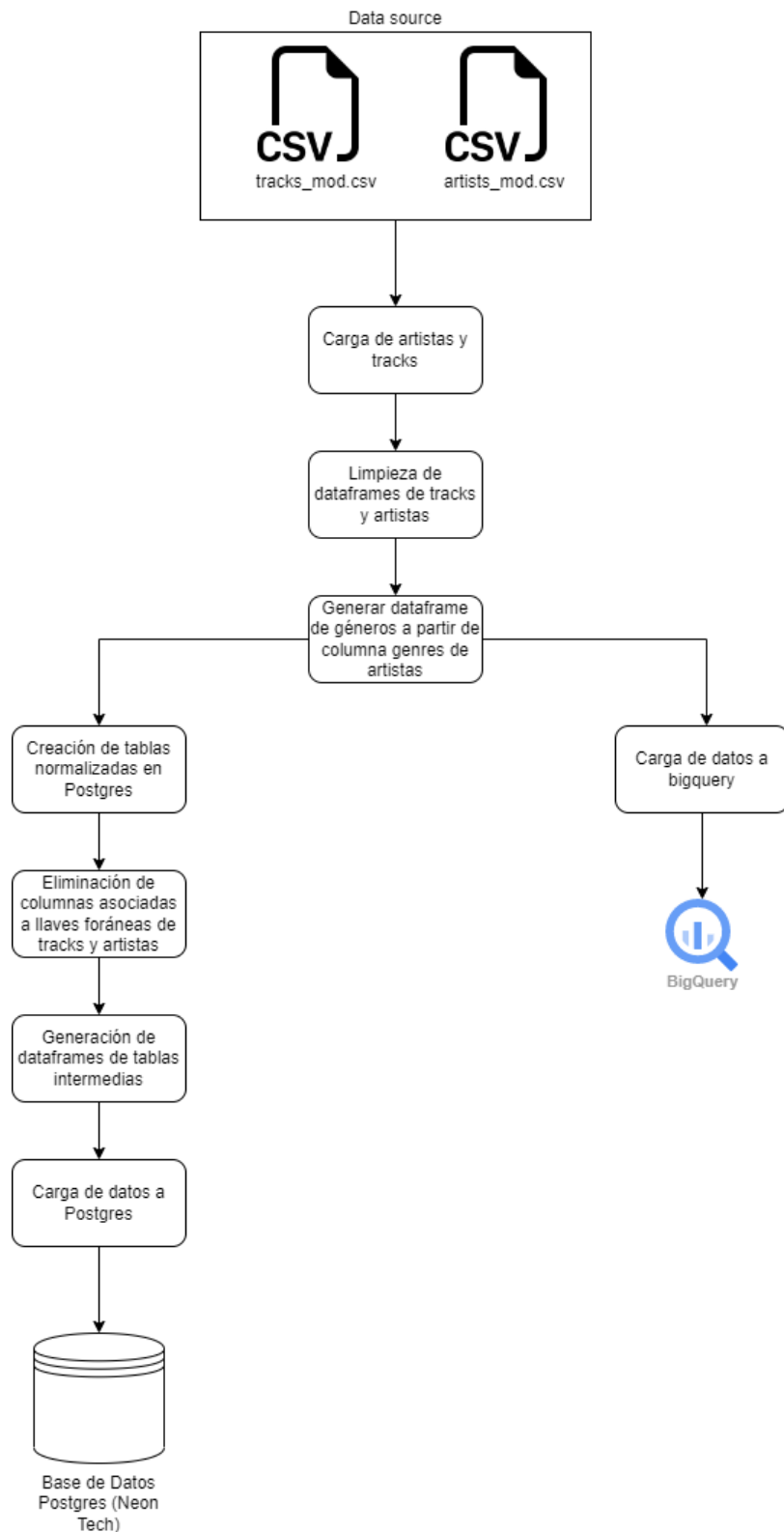
- **Artist**: Se realizó limpieza y asignación de llave primaria. Además de remover columna “genres” y representarla mediante la relación “artist_x_genre”.
- **Track**: Se realizó limpieza y asignación de llave primaria. Además de remover columnas “artists” y “id_artists” y representarlas mediante la relación “artist_x_track”.
- **Genre**: Se crean todos los géneros a partir de la columna “genres” de artist, agregando como llave primaria un id autoincremental.
- **Artist_X_Track**: Se genera a partir de validar si el id de un usuario aparece en la lista de artistas de un track, de manera que sus llaves foráneas son hacia artist_id y track_id.
- **Artist_X_Genre**: Se genera a partir de validar si el id de un usuario aparece en la lista de artistas de un track, de manera que sus llaves foráneas son hacia artist_id y track_id.

5. Esquema de bodega de Datos (BigQuery)

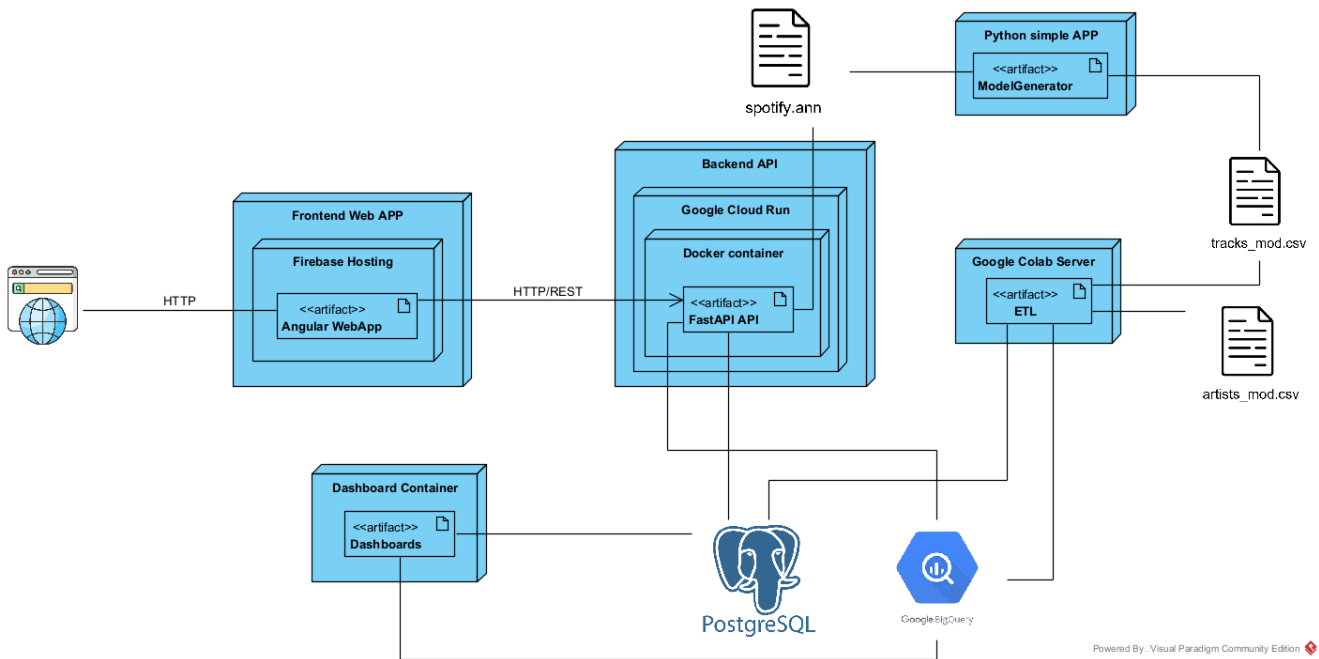


El esquema usado en bigquery, contiene exactamente la misma estructura recibida de parte de las fuentes de datos iniciales. La inserción se realiza después de la limpieza de datos propuesta en las primeras secciones.

6. Arquitectura de ETL



7. Arquitectura General



Para la arquitectura general se cuenta con los siguientes componentes:

Fuentes de datos principales

- **tracks_mod.csv:** Archivo con las tracks sin procesar.
- **artists_mod.csv:** Archivo con los artistas sin procesar.
- **spotify.ann:** Archivo con el modelo predictivo generado por el “ModelGenerator”:
- **PostgreSQL:** Base de datos relacional con los datos limpios y normalizados.
- **BigQuery:** Data warehouse con los datos limpios y sin normalizar.

Componentes principales

- **Google Colab Server:** El servidor de Colab contiene la ETL, en la cual, se cargan los datos a partir de los files de artists y tracks, se procesan, y se almacenan tanto en la base de datos Postgres, como en BigQuery.
- **Python Simple APP:** Esta aplicación contiene el ModelGenerator, se alimenta a partir del archivo de tracks, y genera el modelo de vectores para encontrar las canciones recomendadas. Su salida se almacena en el archivo “spotify.ann”, y se usa posteriormente por la API para devolver recomendaciones a los usuarios.
- **Dashboard Container:** Allí se alojan los dashboards generados a partir de la información almacenada en BigQuery y Postgres.
- **Frontend Web APP:** Este contenedor almacena la aplicación web desarrollada en Angular, la cual se encuentra desplegada haciendo uso del hosting de Firebase. Es accedida a través de la web, y se comunica con el API a través del protocolo REST.
- **Backend API:** Es el componente central del sistema, hace uso de la información almacenada en Postgres y Bigquery para proveer de data al Frontend. Se encuentra desarrollada haciendo uso de FastAPI. Para su despliegue, se construye un

contenedor a través de Dockerfile, y este último es desplegado haciendo uso de Google Cloud Run. Para proveer de la data basada en las 10 canciones recomendadas a un usuario, se usa el archivo "spotify.ann" generado por el ModelGenerator.

8. Descripción del sistema de recomendación

El sistema de recomendación se encuentra implementado a partir de la librería "annoy" para Python. Esta librería se enfoca en la construcción de índices de manera que se puedan encontrar vectores similares en grandes conjuntos de datos.

Las variables utilizadas en este caso para la ejecución del modelo son las siguientes: "explicit", "danceability", "energy", "loudness", "speechiness", "acousticness", "liveness", "valence". La elección de estas variables se basa en que son variables que, al ser cuantitativas, facilitan la comparación entre tracks, pues, entre más similitud entre este conjunto de variables, posiblemente los tracks serán similares, y por ende podrían agradar al oyente.

El proceso de generación del modelo es el siguiente: inicialmente se cargan los tracks limpios. Posteriormente, se genera un índice euclidiano. Una vez generado se agrega al modelo sólo el índice y un dataframe con las variables mencionadas. Finalmente se genera el modelo y se almacena en el archivo "spotify.ann".

Una vez generado el modelo, se traslada el archivo a los recursos del API, allí, en el endpoint /prediction/{track_id}, se usa dicho modelo para calcular 10 recomendaciones de canciones a partir del id de un track. La primera recomendación siempre será la misma canción seleccionada, debido a que el vector será coincidente 100%. A continuación algunos screenshots con ejemplos.











[Volver al home](#)

Detalle de canción



Poesía Acústica #7: Céu Azul
Artistas: [Pineapple StormTV, MC Hariel, Negra Li, Ducon, MC Kevin o Chris, Chris MC, Manuê, Dk 47, Vitão, Salve Malak]
Duración: 9 minutos

Canciones sugeridas

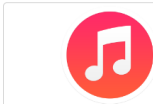
 <div>Poesía Acústica #7: Céu Azul Duración 9 minutos [Pineapple StormTV, MC Hariel, Negra Li, Ducon, MC Kevin o Chris, Chris MC, Manuê, Dk 47, Vitão, Salve Malak]</div>	 <div>Miesta Duración 3 minutos [Neriel]</div>	 <div>ziarnko do ziarnka Duración 3 minutos [chillwagon]</div>	 <div>Art.140 Duración 4 minutos [NotteNera]</div>	 <div>Overkill Duración 3 minutos [Chillnir]</div>
 <div>Two Dope Boyz (In a Cadillac) Duración 2 minutos [Outkast]</div>	 <div>Strazile Duración 4 minutos [B.U.G. Mafia, Mario V]</div>	 <div>Kinky Faster Duración 2 minutos [Suspekt]</div>	 <div>Check Out Time Duración 4 minutos [2Pac]</div>	 <div>Don't Forget To Pray Duración 3 minutos [AKA, ANATHT]</div>

Detalle de canción



El Perdedor
Artistas: [Maluma]
Duración: 3 minutos

Canciones sugeridas



El Perdedor
Duración 3 minutos
[Maluma]



La Chaperona
Duración 2 minutos
[Sonora Cienfuegos]



Por Perro
Duración 4 minutos
[Sebastian Yatra, Luis Figueroa, Lary Over]



Thoughts
Duración 3 minutos
[Michael Carreon]



Por Perro
Duración 4 minutos
[Sebastian Yatra, Luis Figueroa, Lary Over]



Por Perro
Duración 4 minutos
[Sebastian Yatra, Luis Figueroa, Lary Over]



Sola
Duración 3 minutos
[Manuel Turizo]



Sola
Duración 3 minutos
[Manuel Turizo]



Sola
Duración 3 minutos
[Manuel Turizo]



La Curita
Duración 3 minutos
[Los Mello]

9. Descripción de los Dashboards

Para los Dashboards, se realizó la elección de 6 KPIs acompañados de algunas gráficas desarrolladas en PowerBI. Dichos gráficos se encuentran localizados en el repositorio de GitHub asociado a las referencias, en la carpeta de Dashboards.

Los KPIs seleccionados son los siguientes:

Los dos primeros KPIs se encuentran respaldados por el Dashboard de “followers_by_artist”:

- **Número de seguidores:** Este KPI mide la cantidad total de seguidores que tiene cada artista. Se puede medir el número total de seguidores para cada artista, y también el número promedio de seguidores entre todos los artistas. Su importancia para el negocio se destaca en que, al conocer los artistas que puedan tener mayor número de seguidores, es posible para la compañía buscar exclusivas con dichos artistas, u ofrecer diferentes contratos, debido a que al tener más seguidores, quizás sus reproducciones sean más altas, y por ende más ganancias para la compañía.
- **Tasa de Crecimiento de Seguidores:** Este KPI mide la tasa de crecimiento de seguidores para cada artista durante un período específico, por ejemplo, el crecimiento de seguidores durante un mes o un año. Se puede calcular como la diferencia en porcentaje entre el número actual de seguidores y el número de seguidores al inicio del período, dividido por el número de seguidores al inicio del período. Su importancia en el negocio, se basa en que, es posible establecer diferentes metas para los artistas con mejores tasas, de manera que, al ofrecerles contratos basados en las metas, será posible conseguir exclusivas, y establecer líneas de trabajo que permitan estar sobre la competencia.

Los dos siguientes KPIs se respaldan por el gráfico de “popularity_by_artists”:

- **Índice de Popularidad:** Este KPI mide la popularidad de cada artista en comparación con los demás artistas. Se puede medir el índice de popularidad de

cada artista utilizando el número de reproducciones, visitas o descargas que han obtenido sus canciones o videos, y compararlo con el índice de popularidad de los demás artistas para identificar quiénes son los más populares.

- **Número de Canciones por Artista:** Este KPI mide la cantidad de canciones que ha lanzado cada artista. Este KPI puede ser importante para evaluar el nivel de productividad y compromiso de cada artista, así como también para identificar a los artistas más prolíficos y los que tienen un menor nivel de producción.

La importancia de negocio de los dos KPIs anteriores va en conjunto. Al conocer el número de canciones por artista, y compararlo vs su índice de popularidad, es posible constatar si realmente ese índice de popularidad va asociado a su cantidad de canciones, o, si por el contrario es realmente por la popularidad del artista en las pocas canciones que tenga, lo que va a permitir tomar decisiones tales como, pagarle a un artista popular independiente de la cantidad de canciones que tenga, o si pagarle según el impacto real que tengan sus canciones, de manera que sea posible optimizar los contratos existentes.

Los últimos dos KPIs se encuentran soportados por el gráfico “popularity_by_name”:

- **Índice de Popularidad:** Este KPI mide la popularidad de cada canción en comparación con las demás canciones. Se puede medir el índice de popularidad utilizando el número de reproducciones, visitas o descargas que ha obtenido cada canción, y compararlo con el índice de popularidad de las demás canciones para identificar cuáles son las más populares.
- **Porcentaje de Reproducciones por Canción:** Este KPI mide el porcentaje de reproducciones que ha obtenido cada canción en relación con el total de reproducciones de todas las canciones. Este KPI puede ser importante para identificar las canciones que están recibiendo una mayor atención de los usuarios en comparación con las demás.

Tal como los anteriores KPIs, estos dos se relacionan para poder ser entendidos en conjunto. Conociendo la cantidad de reproducciones por canción y el índice de popularidad, es posible definir los tipos de contratos con los artistas, siguiendo un modelo de exclusividad, en donde un artista lance sus canciones como primicia en la plataforma, si tengo garantía de que sus canciones tiene la popularidad alta. Mientras que con otros artistas puedo hacerlo dependiendo de el impacto real que tengan.

10. Referencias

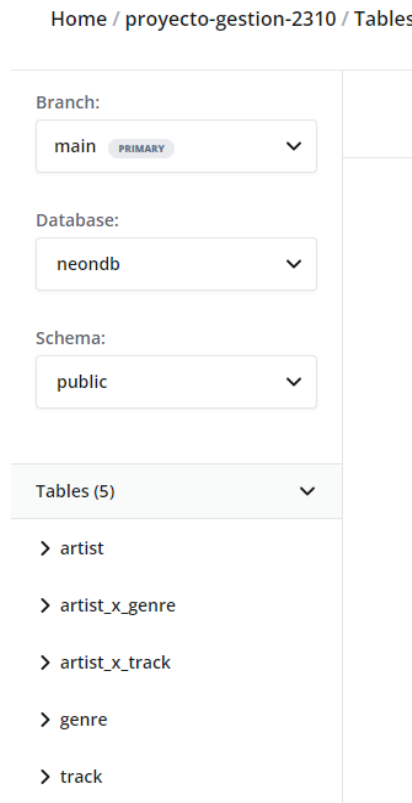
- **Github:** <https://github.com/miguelgutierrez/proyecto-gestion-datos-2310>
- **Spotify API:** <https://developer.spotify.com/documentation/web-api/reference>
- **Neon Tech:** <https://neon.tech/>

11. Anexos

A continuación se presentan los anexos asociados a las pruebas de correcta ejecución en inserción de datos mediante el script usado.

Creación de tablas e inserción de datos en Postgres

Para el trabajo con Postgres, se creó una instancia gratuita de base de datos Postgres con la ayuda de neon tech, de manera que fuese posible acceder a dicha instancia desde el Notebook realizado con Google Colab.



Vemos en la imagen anterior la existencia de las tablas creadas en la instancia online de la base de datos.

La inserción de datos se puede evidenciar en los resultados del notebook de Colab, sin embargo se adjuntan los pantallazos

```
[ ] query_tracks = "SELECT * FROM track LIMIT 5"
print(conn.execute(query_tracks).fetchall())

[('35iwgR4jXetI318WEwsa1Q', 'Carve', Decimal('6.0'), 126903, 0, datetime.date(1922, 2, 22),

[ ] query_artists = "SELECT * FROM artist LIMIT 5"
print(conn.execute(query_artists).fetchall())

[('0DheY5irMjBUeLybbCUEZ2', Decimal('0.0'), 'Armid & Amir Zare Pashai feat. Sara Rouzbehani', 0),
```

```
[ ] query_genres = "SELECT * FROM genre LIMIT 5"
print(conn.execute(query_genres).fetchall())

[(1, '21stcenturyclassical'), (2, '432hz'), (3, '48g'), (4, '8-bit'), (5, '8d')]
```

```
[ ] query_artist_x_genre = "SELECT * FROM artist_x_genre LIMIT 5"
print(conn.execute(query_artist_x_genre).fetchall())

[(('3F0cz0CoQjXRhfnKLqrXQq', 4790), ('02A3cEv1LLCbIMVDrK2GHV', 5027), ('7xx0gYr6iMecpDbSynNzWF', 3901),
```

```
[ ] query_artist_x_track = "SELECT * FROM artist_x_track LIMIT 5"
print(conn.execute(query_artist_x_track).fetchall())

[(('3BiJGZsyX9sJchTqcSA7Su', '1MD00bbza9l0t0Zpgcwagy'), ('58wzyK6DupVsypvs3QV2Fo', '1MD00bbza9l0t0Zpgcwagy'),
```

Así como también pantallazos de las tablas con data en línea

artist			
#	id	followers	name
1	0DheY5irMjBUeLybbCUEZ2	0.0	Armid & Amir Zare Pashai
2	0DlhY15l3wsrnlfgio2bjU	5.0	อนุชา ทรัพย์
3	0DmRESX2JknGPQyO15yXg7	0.0	Sadaa
4	0DmhnbHjm1qw6NCYPeZNgl	0.0	Tra'gruda

track						
#	id	name	popularity	duration_ms	explicit	release_date
1	35iwgR4jXetl318WEWsa1Q	Carve	6.0	126903	0	1922-02-22
2	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0.0	98200	0	1922-06-01
3	07A5yehtSnoedVijAZkNnc	Vivo para Quererte - Remasterizado	0.0	181640	0	1922-03-21

genre		
#	id	name
1	1	21stcenturyclassical
2	2	432hz
3	3	48g
4	4	8-bit

artist_x_genre		
#	artist_id	genre_id
1	3FOcz0CoQjXRhnKLqrXQq	4790
2	02A3cEvLLCbIMVDrK2GHV	5027
3	7xx0gYr6IMecpDbSynNzWF	3901

artist_x_track		
#	artist_id	track_id
1	3BijGZsyX9sjchTqcSA7Su	1MD00bbza9l0t0Zpgcwagy
2	58wzyK6DupVsypvs3QV2Fo	1MD00bbza9l0t0Zpgcwagy
3	6lXiGaWjISZnER53Zje6QO	1MD00bbza9l0t0Zpgcwagy
4	3BijGZsyX9sjchTqcSA7Su	1O9iZyzufN1fUdVO97mmm5

Inserción de datos en BigQuery

Para la inserción de datos en BigQuery se creó un proyecto inicial llamado proyecto-gestion-2310 con un esquema llamado GESTION. En donde se insertaron los datos, cuya evidencia se puede ver en el Notebook, pero se adjuntan los pantallazos igualmente

▼	proyecto-gestion-2310	☆	⋮
▶	↻ Conexiones externas		⋮
▼	🗃 GESTION	☆	⋮
	🗃 ARTISTS_icjg	☆	⋮
	🗃 GENRES_buak	☆	⋮
	🗃 TRACKS_mdfh	☆	⋮

```
query = f"""SELECT * FROM `proyecto-gestion-2310.{BQ_TABLE_NAME}` LIMIT 5"""
pd.read_gbq(query, credentials=credentials)
```

	id	followers	genres	name	popularity
0	7frYUe4C7A42uZqCzD34Y4	53636.0	['desi pop', 'punjabi hip hop', 'punjabi pop']	Sultaan	53
1	6acbdy69rtlv8m9EW31MYI	72684.0	['afro dancehall', 'afropop', 'azontobeats', '...]	Phyno	51
2	72578usTM6Cj5qWsi471Nc	248568.0	['filmi', 'indian folk', 'indian rock', 'kanna...]	Raghu Dixit	52

```
[ ] query = f"""SELECT * FROM `proyecto-gestion-2310.{TRACKS_TABLE_NAME}` LIMIT 5"""
pd.read_gbq(query, credentials=credentials)
```

	id	name	popularity	duration_ms	explicit
0	0tb4r8iCxRL7zwdwVD4vF0	La Gioconda: Act III: O madre mia, nell'isola ...	0.0	112907	0
1	5fxTsaLKOWt1cZsr7sbwBZ	La Gioconda: Act IV: Suicidio!	0.0	274387	0
2	1VBKpTI3NNOOb6KrcJYbMQ	La Gioconda: Act I: L'ora non giunse ancor	0.0	170120	0

```
query = f"""SELECT * FROM `proyecto-gestion-2310.{GENRES_TABLE_NAME}` LIMIT 5"""
pd.read_gbq(query, credentials=credentials)
```

	id	name
0	1	21stcenturyclassical
1	2	432hz
2	3	48g
3	4	8-bit
4	5	8d

Igualmente se adicionan pantallazos de BigQuery

ARTISTS_icjg						
CONSULTA COMPARTIR COPIAR INSTANTÁNEA BORRAR EXPORT						
ESQUEMA		DETALLES		VISTA PREVIA		
Fila	id	followers	genres	name	popularity	
1	7frYUe4C7A42uZqCzD34Y4	53636.0	['desi pop', 'punjabi hip hop', 'pu...	Sultaan	53	
2	6acbdy69rtlv8m9EW31MYI	72684.0	['afro dancehall', 'afropop', 'azontobeats', 'nigerian hip hop', 'nigerian pop']	Phyno	51	
3	72578usTM6Cj5qWsi471Nc	248568.0	['filmi', 'indian folk', 'indian rock', ...	Raghu Dixit	52	
4	4rk6HLvoZhLFUTcUhG9WfC	5644.0	[]	Deacon	52	

TRACKS_mdffh						
CONSULTA COMPARTIR COPIAR INSTANTÁNEA BORRAR EXPORTAR						
ESQUEMA		DETALLES		VISTA PREVIA		
Fila	id	name	popularity	duration_ms	explicit	artists
1	0tb4r8ICxRL7zwdwVD4vF0	La Gioconda: Act III: O madre ...	0.0	112907	0	['Amilcare Ponchielli', 'Maria Callas', 'Gianni Poggi', 'Paolo Silveri', 'Fedora Barbieri', 'Giulio Neri', 'Maria Amadini', 'RAI Chorus,

GENRES_buak		
CONSULTA		
ESQUEMA		DETALLES
Fila	id	name
1	1	21stcenturyclassical
2	2	432hz
3	3	48g
4	4	8-bit
5	5	8d

Finalmente, para la ejecución correcta del script, es posible remitirse al notebook referenciado en la sección de referencias de este documento. En donde se encuentra el output de los pasos ejecutados en el script.

Identificación de KPIs y gráficos: