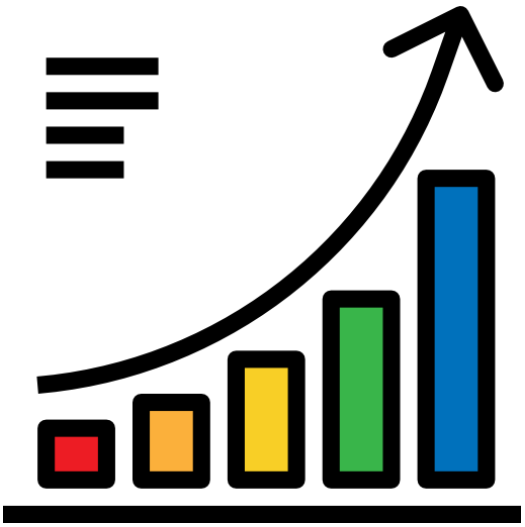


Spotify Stats

con BigQuery y Data Studio

Pablo González

spotifystats [@] gonzaleztroyano.es

Junio 2021 - Modificado Octubre 2021

Índice

Abstract **3**

Datos **4**

Acceder a los datos 4

Transformación de datos 5

Transformación de datos - método 1 6

Transformación de datos - método 2 8

Otros comandos interesantes 9

Alojamiento de datos 11

Cargar los datos 11

Visualización de datos e informes **13**

DS: Página principal y resumen 13

DS: Gráfico de rectángulos, simple 15

DS: Gráfico de rectángulos, comparativa 16

DS: Visión de tendencias 17

Acceder al informe 18

Abstract

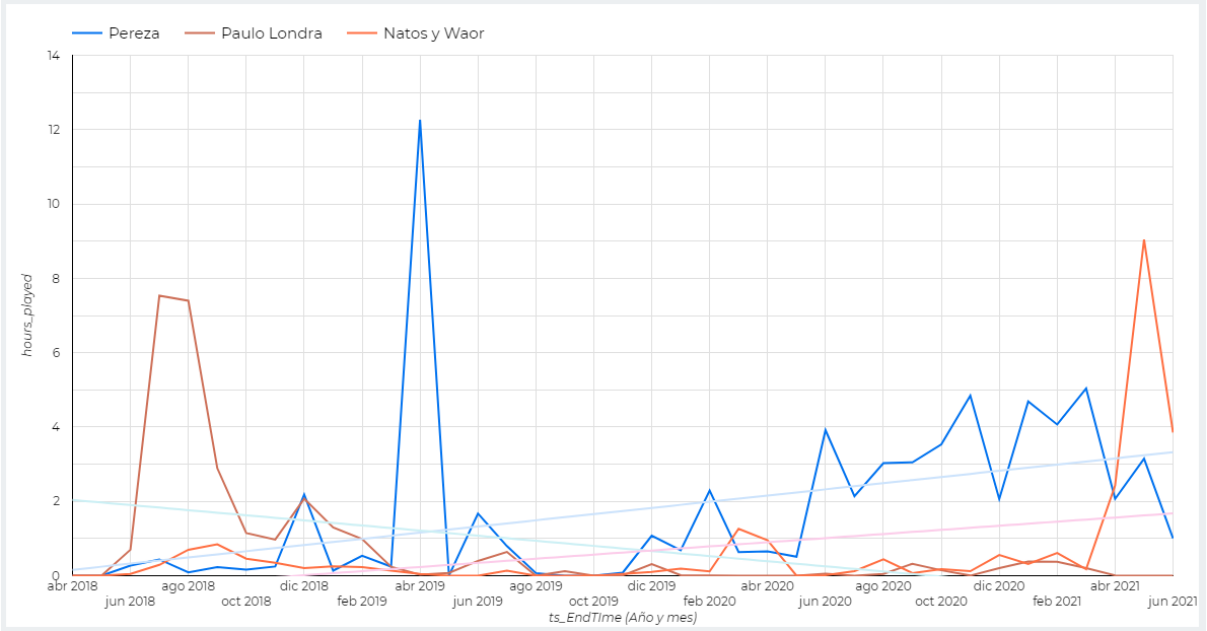
Si una empresa domina a día de hoy el mercado de la reproducción de música es Spotify[1]. Con un crecimiento exponencial, tanto en usuarios absolutos[2] como en usuarios de pago[3].

Uno de esos usuarios es un servidor, que lleva usando Spotify desde hace años y es usuarios de pago desde hace más de un año, que coinciden con el crecimiento exponencial de las horas de reproducción.

¿Y qué ocurre cuando se juntan estadísticas, bases de datos, *data visualization,* e inquietudes informáticas? Nada sencillo, desde luego. Pero también puede surgir una gran idea y sobre todo, mucho aprendizaje.

En este documento se pretende recoger el proceso y el resultado de crear una panel de visualización de estadísticas de reproducción en dicha plataforma. Navegando por Internet es posible encontrar multitud de ejemplos[4] [5], incluso utilizando la misma herramienta que utilizo en mi caso, Google Data Studio aunque con ciertos cambios en el tratamiento de los archivos.

Aquí tenemos un ejemplo de la evolución de las horas escuchadas para tres artistas diferentes. No han sido elegidos al azar, sino de forma consciente para ver las líneas de tendencia.



[1] Statista • Music streaming market share
[2] Statista • Spotify MAUs worldwide from 1st quarter 2015 to 1st quarter 2021
[3] Statista • Spotify premium subscribers worldwide from 1st quarter 2015 to 1st quarter 2021
[4] Get Your Spotify Streaming History With Python | by Vlad Gheorghe
[5] A simple guide to visualizing your Spotify listening data... badass-ly

Datos

Entre muchos otros detalles beneficiosos que la entrada en vigor del RGPD (Régimen General de Protección de Datos; GDPR en inglés) trajo consigo fue la garantización del acceso a los datos por parte de los usuarios.

¿Qué datos guarda Spotify? Pues realmente guarda muchísima información. Pero es realmente transparente en este sentido y es una empresa europea, por lo que *a priori* lo hace con garantías.

En esta página[6] se puede obtener más información sobre la protección de datos en la Unión Europea, en esta página[7] se encuentra la información concreta de Spotify en lo que se refiere a datos y privacidad.

Acceder a los datos

Para acceder a la información, lo primero que debemos tener es una cuenta de Spotify (esto se sobreentiende pues no es posible tener un historial de reproducción si no tenemos cuenta en la plataforma).

Según la cantidad de datos a los que queramos acceder tenemos dos vías. Por un lado, tenemos la vía "sencilla". Esta es la opción recomendada para la mayoría de los usuarios, pues permite "jugar" con datos reales de forma sencilla. Mediante esta opción tenemos la posibilidad de descargar, entre otra información, nuestro historial de reproducción del último año. El detalle de la información contenida en esta solicitud se encuentra disponible en este enlace[8]. Podemos solicitar dicha información desde el apartado de privacidad de nuestra cuenta de Spotify, accesible a través de este enlace[9]. Debemos seguir los pasos indicados en la parte inferior de la página, verificar por correo electrónico que deseamos los datos y en cuestión de unos días (3 en mi caso, pero legalmente la demora puede ser de hasta 30 días) recibiremos una notificación vía correo electrónico con un enlace para descargar un archivo en formato comprimido ZIP con nuestros datos (son divididos según la categoría en archivos JSON) y un magnífico PDF donde se explica cómo interpretar la información.

Si necesitamos información adicional, nuestras reproducciones del último año no son suficientes y/o queremos todo (todo, todo) el registro técnico que la empresa guarda sobre nosotros debemos contactar con el equipo de Soporte de Spotify utilizando este formulario[10] o enviar un correo electrónico a la dirección privacy@spotify.com. Esta otra vía es algo más lenta, serán necesarias varias comunicaciones por parte nuestra y del equipo

[6] Protección de datos | Comisión Europea
[7] Data rights and privacy settings - Spotify
[8] Understanding my data - Spotify
[9] Account Settings - Privacy
[10] Contact Customer Support - Spotify

de Spotify. Recibiremos por un lado la información técnica detallada y días más tarde todo el historial de reproducciones de la cuenta (ordenado de forma extraña, aparentemente).

Esta segunda forma de acceder nos provee de quizá demasiada información, pero sin duda compensa símplemente por el hecho de conocer la magnitud de la recogida de datos.

También tenemos disponible la web de estadísticas a nivel global de Spotify[11] que nos provee de tendencias de escucha muy interesantes. Aquí se muestra un ejemplo con el programa de visualización de datos *Tableau*[12] y en este enlace[13] se puede acceder a un informe realizado con dicho programa con la información de Spotify Charts.

Transformación de datos

Según accedamos a los datos utilizando una vía u otra, el esquema y la magnitud será distinta. En el caso de el primer método, con el historial de 1 año, un ejemplo de datos podría ser el siguiente (fragmento real ligeramente editado):

```
[
  {
    "endTime" : "2020-03-01 14:17",
    "artistName" : "El Canto Del Loco",
    "trackName" : "No Puedo Vivir Sin Ti",
    "msPlayed" : 32004
  },
  {
    "endTime" : "2020-03-02 19:36",
    "artistName" : "Arcangel",
    "trackName" : "Mi Primer Kilo",
    "msPlayed" : 21152
  },
  {
    "endTime" : "2020-03-03 12:54",
    "artistName" : "Shawn Mendes",
    "trackName" : "There's Nothing Holdin' Me Back",
    "msPlayed" : 2586
  },
  {
    "endTime" : "2020-03-04 06:28",
    "artistName" : "Sia",
    "trackName" : "Thunderclouds (feat. Sia, Diplo, and Labrinth)",
    "msPlayed" : 187026
  }
]
```

[11] Spotify Charts
[12] Cómo visualizar las tendencias de la música de Spotify en Tableau
[13] https://public.tableau.com/app/profile/scott.teal/viz/SpotifyWebinar/Dashboard1

Si accedemos a los datos utilizando el segundo método, los datos no están tan procesados, por lo que recibimos JSON (uno o más, dependiendo de cuánto hayamos utilizado el servicio). Una entrada de dicho archivo es así:


```
{"ts":"2018-09-07T06:28:05Z","username":"XXXXXXXXX","platform":"Android
OS 8.1.0 API 27 (MARCA, MODELO)","ms_played":0,"conn_country":"ES",
"ip_addr_decrypted":"XX.XX.XX.XX","user_agent_decrypted":"XXXX","master_
metadata_track_name":"Perdóname","master_metadata_album_artist_name":"FM
K","master_metadata_album_album_name":"Perdóname","spotify_track_uri":"s
potify:track:4Uy3Q3h0VbWWyyiB1MmRoP","episode_name":null,"episode_show_n
ame":null,"spotify_episode_uri":null,"reason_start":"clickrow","reason_e
nd":"unexpected-exit-while-paused","shuffle":true,"skipped":null,"offlin
e":false,"offline_timestamp":1536296766003,"incognito_mode":false}
```


Nótese que se han reemplazado ciertos datos por equis. En mi caso, recibí 5 archivos endsong_0.json, endsong_1.json ... endsong_4.json

Con más de 70.000 registros (siendo aproximadamente la mitad del último año).

Si bien existen aplicaciones que permiten trabajar directamente con esta clase de archivos sin "trabajarlos" mucho, en mi caso han sido necesarias ciertas transformaciones.

Al ser archivos en formato JSON, trabajaremos bajo Linux (Ubuntu 18.04 en Windows Subsystem for Linux[14] y Debian en Google Cloud Shell[15]) con el comando jq[16], entre otros comandos consideramos de uso general.

Transformación de datos - método 1

Si los datos los hemos obtenido utilizando el método 1, tendremos X archivos del estilo StreamingHistoryX.json, siendo X un número incremental empezando por 0 la sucesión.

```
cat StreamingHistory0.json | jq -c '.[]' > StreamingHistory0_mod.json
cat StreamingHistory1.json | jq -c '.[]' > StreamingHistory1_mod.json
cat StreamingHistory2.json | jq -c '.[]' > StreamingHistory3_mod.json
```

Con el comando anterior lo que hacemos es convertir cada registro en una línea, además, eliminamos los corchetes de inicio y cierre del JSON.
Realizamos el proceso archivo por archivo para poder detectar errores. En este caso no son archivos pesados (~1.5MB) pero puede sernos útil en ciertas ocasiones.

[14] Instalación de WSL en Windows 10
[15] Google Cloud Shell
[16] jq is a lightweight and flexible command-line JSON processor

Veamos la entrada y la salida de este comando.

De los datos arriba indicados como ejemplo

```
[
  {
    "endTime" : "2020-03-01 14:17",
    "artistName" : "El Canto Del Loco",
    "trackName" : "No Puedo Vivir Sin Ti",
    "msPlayed" : 32004
  },
  {
    "endTime" : "2020-03-02 19:36",
    "artistName" : "Arcangel",
    "trackName" : "Mi Primer Kilo",
    "msPlayed" : 21152
  },
  {
    "endTime" : "2020-03-03 12:54",
    "artistName" : "Shawn Mendes",
    "trackName" : "There's Nothing Holdin' Me Back",
    "msPlayed" : 2586
  },
  {
    "endTime" : "2020-03-04 06:28",
    "artistName" : "Sia",
    "trackName" : "Thunderclouds (feat. Sia, Diplo, and Labrinth)",
    "msPlayed" : 187026
  }
]
```

Obtenemos lo siguiente:

```
{"endTime":"2020-03-01 14:17","artistName":"El Canto Del Loco","trackName":"No Puedo Vivir Sin Ti","msPlayed":32004}
{"endTime":"2020-03-02 19:36","artistName":"Arcangel","trackName":"Mi Primer Kilo","msPlayed":21152}
{"endTime":"2020-03-03 12:54","artistName":"Shawn Mendes","trackName":"There's Nothing Holdin' Me Back","msPlayed":2586}
{"endTime":"2020-03-04 06:28","artistName":"Sia","trackName":"Thunderclouds (feat. Sia, Diplo, and
Labrinth)","msPlayed":187026}
```

Si bien es menos legible a primera vista, para cualquier software será sencillo reconocer cada entrada al existir un salto de línea separándolas. ¿Es un JSON correctamente formado? No al completo, pero para nuestro caso de uso es justo lo que necesitamos.

Por sencillez, es muy recomendable unir todos los archivos en uno, que será el que utilicemos posteriormente.

¿Cómo podemos unificar los archivos? ¡Con comandos de Linux!

```
cat StreamingHistory0_mod.json > StreamingHistory.json
cat StreamingHistory1_mod.json >> StreamingHistory.json
cat StreamingHistory2_mod.json >> StreamingHistory.json
```

Con estos comandos estamos redirigiendo la salida del "cat" hacia el archivo *StreamingHistory.json*. En la segunda y siguientes ejecuciones, utilizamos el doble direccionador ">>" para anexar al final del archivo la salida, en caso de utilizar solo uno estaríamos sustituyéndolo.

Una vez hayamos completado estos pasos tendremos un archivo único con todo nuestro historial del último año.

Transformación de datos - método 2

Si hemos decidido que queremos obtener el historial completo de nuestra reproducción en Spotify, puede ser más complicado el proceso. No realmente por los comandos en sí, sino porque tenemos más registros (más tiempo de historial) y estos registros son más extensos al contener más datos. En mi caso, recibí 5 archivos JSON por parte de Spotify. 4 de 10MB (aprox) y un último de 5MB.

```
cat endsong_0.json | jq  -c '.[]' > endsong_0_mod.json
cat endsong_1.json | jq  -c '.[]' > endsong_1_mod.json
cat endsong_2.json | jq  -c '.[]' > endsong_2_mod.json
cat endsong_3.json | jq  -c '.[]' > endsong_3_mod.json
cat endsong_4.json | jq  -c '.[]' > endsong_4_mod.json
```

Con el comando anterior (al igual que hemos hecho anteriormente) lo que hacemos es convertir cada registro en una línea, además, eliminamos los corchetes de inicio y cierre del JSON.
Realizamos el proceso archivo por archivo para poder detectar errores. En este caso no son archivos pesados (~1.5MB) pero puede sernos útil en ciertas ocasiones.

Para terminar, y como hemos realizado con los datos al obtenerlos mediante el método 1, lo mejor es combinar los archivos en un único archivo conjunto. Esto nos permitirá flexibilidad y optimización a la hora de cargar los datos en BigQuery. Sí es cierto que en ciertos supuestos puede ser contraproducente tener un único archivo, si se van a producir lecturas simultáneas o aleatorias. De hecho, el archivo detallado (método 2) no está ordenado por fecha de reproducción.

Para unir los archivos es suficiente con ejecutar los siguientes comandos

```
cat endsong0_mod.json > endsong_completo.json
cat endsong_1_mod.json >> endsong_completo.json
cat endsong_2_mod.json >> endsong_completo.json
cat endsong_3_mod.json >> endsong_completo.json
cat endsong_4_mod.json >> endsong_completo.json
```

Con estos comandos estamos redirigiendo la salida del "cat" hacia el archivo *endsong_completo.json*. En la segunda y siguientes ejecuciones, utilizamos el doble direccionador ">>" para anexar al final del archivo la salida, en caso de utilizar solo uno estaríamos sustituyéndolo.

Otros comandos interesantes

Los comandos aquí citados son muy "low level". Con esto se quiere indicar que no son comandos especializados para esta tarea, sino todo lo contrario. Conocer estos comandos nos va a permitir ser más eficientes en nuestro día a día.

No incluimos en este apartado los comandos `cat` ni `jq` al haber sido comentados anteriormente.

Se ha utilizado el comando `head`. Que permite mostrar la parte superior (entiéndase como superior la parte inicial de un archivo) de los documentos de texto (como JSON). Por defecto si solo indicamos `head %archivo%` nos muestra las 10 primeras líneas de %archivo%. En el caso del método 1 puede ser suficiente, pero en el método 2 no tenemos saltos de línea en el archivo, todos los datos forman parte de un *stream* de caracteres. Para modificar la cantidad de datos que ofrece como salida este comando debemos utilizar los modificadores `-n X` para indicar el número de líneas que se mostrarán, o `-c X` siendo X el número de bytes que se mostrarán.

```
pablo@cloudshell:~$ head -n 2 StreamingHistory1_mod.json
{"endTime":"2020-08-28 15:06","artistName":"Amaral","trackName":"El universo sobre mí","msPlayed":249704}
{"endTime":"2020-08-28 15:10","artistName":"Maldita Nerea","trackName":"Tu Mirada Me Hace Grande","msPlayed":257626}
pablo@cloudshell:~$ head -n 2 StreamingHistory0_mod.json
{"endTime":"2020-03-01 14:17","artistName":"El Canto Del Loco","trackName":"No Puedo Vivir Sin Ti","msPlayed":32004}
{"endTime":"2020-03-02 19:36","artistName":"Arcangel","trackName":"Mi Primer Kilo","msPlayed":21152}
```

Si el comando `head` muestra la parte inicial de un archivo, el comando tail muestra las últimas líneas de un archivo. Junto con head, nos permitirá comprobar qué información contiene los archivos sin necesidad de abrirlos con un editor de texto (gráfico o no) ni imprimir por pantalla el contenido completo, con la consiguiente mejora de eficiencia.

Este comando funciona con los mismos modificadores de comandos que head. Utilizaremos `-n X` para indicar el número de líneas que se mostrarán, o `-c X` siendo X el número de bytes que se mostrarán.

```
pablo@cloudshell:~$ tail -n 2 StreamingHistory0_mod.json
{"endTime":"2020-08-28 14:59","artistName":"El Canto Del Loco","trackName":"Puede Ser (with Amaia
Montero)","msPlayed":173482}
{"endTime":"2020-08-28 15:02","artistName":"Despistaos","trackName":"Física o química","msPlayed":190149}
pablo@cloudshell:~$ tail -c 300 StreamingHistory0_mod.json
56","artistName":"Pereza","trackName":"Animales","msPlayed":944}
{"endTime":"2020-08-28 14:59","artistName":"El Canto Del Loco","trackName":"Puede Ser (with Amaia
Montero)","msPlayed":173482}
{"endTime":"2020-08-28 15:02","artistName":"Despistaos","trackName":"Física o química","msPlayed":190149}
```

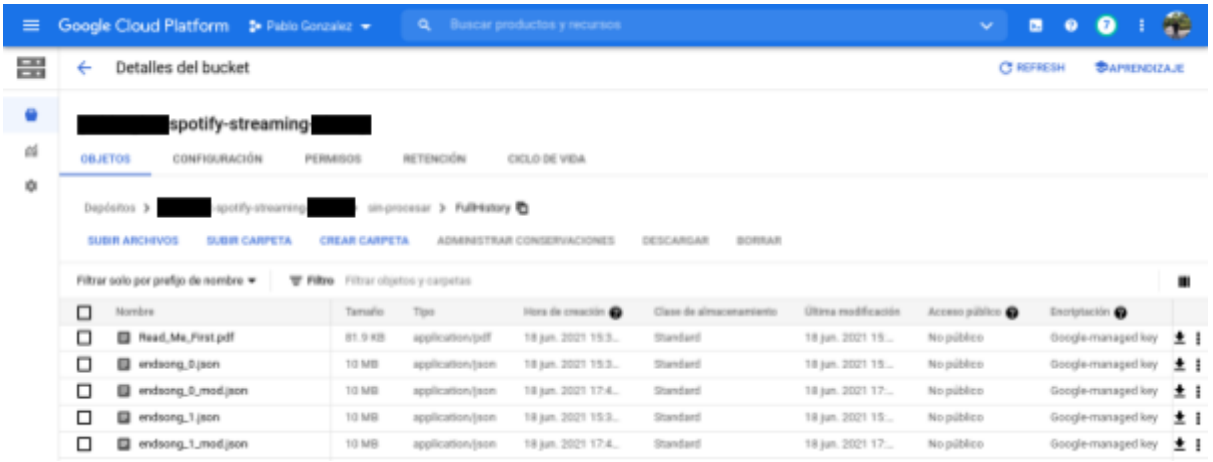
Uno de los problemas con los que nos podemos encontrar es que tenemos muchas información muy compacta y puede ser complicado ver de forma sencilla datos clave.

Para ese filtrado de información tenemos el comando `grep`. Al utilizar el comando destacamos partes de la salida de un comando. En el caso de los archivos obtenidos mediante el método 2 es relativamente importante pues no es posible ver de forma sencilla al ser un único *stream* de datos. Utilizando el siguiente comando se nos destacará en la salida cada vez que aparezca "ts" en color rojo. Se utiliza "ts" pues cada entrada va precedida por ese texto, que marca el *timestamp* del final de la reproducción.

```
pablo@cloudshell:~$ head -c 1000 endsong_4_mod.json | grep ts
{"ts":"2020-08-31T17:10:01Z","username":"Spotify_User","platform":"Android OS 10 API 29 (Xiaomi, Redmi Note
7)","ms_played":176546,"conn_country":"ES","ip_addr_decrypted":"XX.XX.XX.XX","user_agent_decrypted":"unknown","master_m
etadata_track_name":"ily (i love you baby) (feat. Emilee)","master_metadata_album_artist_name":"Surf
Mesa","master_metadata_album_album_name":"ily (i love you baby) (feat.
Emilee)","spotify_track_uri":"spotify:track:62aP9fBQKYKxi7PDXwcUAS","episode_name":null,"episode_show_name":null,"spoti
fy_episode_uri":null,"reason_start":"trackdone","reason_end":"trackdone","shuffle":false,"skipped":null,"offline":true,
"offline_timestamp":1598880113075,"incognito_mode":false}
{"ts":"2018-07-20T21:54:41Z","username":"Spotify_User","platform":"Android OS 7.1.2 API 25 (Xiaomi, Redmi
5A)","ms_played":138060,"conn_country":"ES","ip_addr_decrypted":"XX.XX.XX.XX","user_agent_decrypted":"unknown","master_
metadata_track_name":"Perdóname","master_metadata_album_artist_name"
```

En mi caso, al estar trabajando en Google Cloud Platform[17] y con segmentos de Storage[18] también ha sido muy interesante utilizar el comando `gsutil`[19]. Este comando permite interactuar con archivos y segmentos de Storage como si estuvieran en la máquina conectados. De esta forma podemos copiar los archivos alojados en Storage para trabajar con ellos en "local" (entrecomillo, pues en el caso de Cloud Shell trabajamos en una máquina virtual) y luego volver a subirlos. En la siguiente imagen se puede ver la interfaz de trabajo.

[17] Google Cloud: Cloud Computing Services
[18] Google Cloud Storage
[19] gsutil tool | Cloud Storage



Y el comando gsutil es tan sencillo como el siguiente ejemplo:

```
pablo@cloudshell:~$ gsutil cp \ gs://XXXXXXXX-spotify-streaming-XXXXXXXX/sin-procesar/FullHistory/endsong_4_mod.json
.
Copying gs://XXXXXXXX-spotify-streaming-XXXXXXXX/sin-procesar/FullHistory/endsong_4_mod.json...
/ [1 files][  4.0 MiB/  4.0 MiB]
Operation completed over 1 objects/4.0 MiB.
```

Alojamiento de datos

Ya tenemos disponibles los datos, los hemos tratado de forma que sea sencillo para el *software* trabajar con ellos. Si bien podríamos desplegar un servidor de MySQL o postgresql, tendríamos que gestionar demasiada "infraestructura". Se han elegido aplicaciones Cloud SaaS, Cloud Software as a Service, o más concretamente DBaaS para el alojamiento de los datos.

La herramienta utilizada es BigQuery[20] al ser sencilla, no necesitar de administración, capacidad de almacenamiento al nivel de Petabytes e interfaz SQL. Además, tiene una capa gratuita que permite almacenar 10GB y analizar 1TB al mes sin ningún coste.

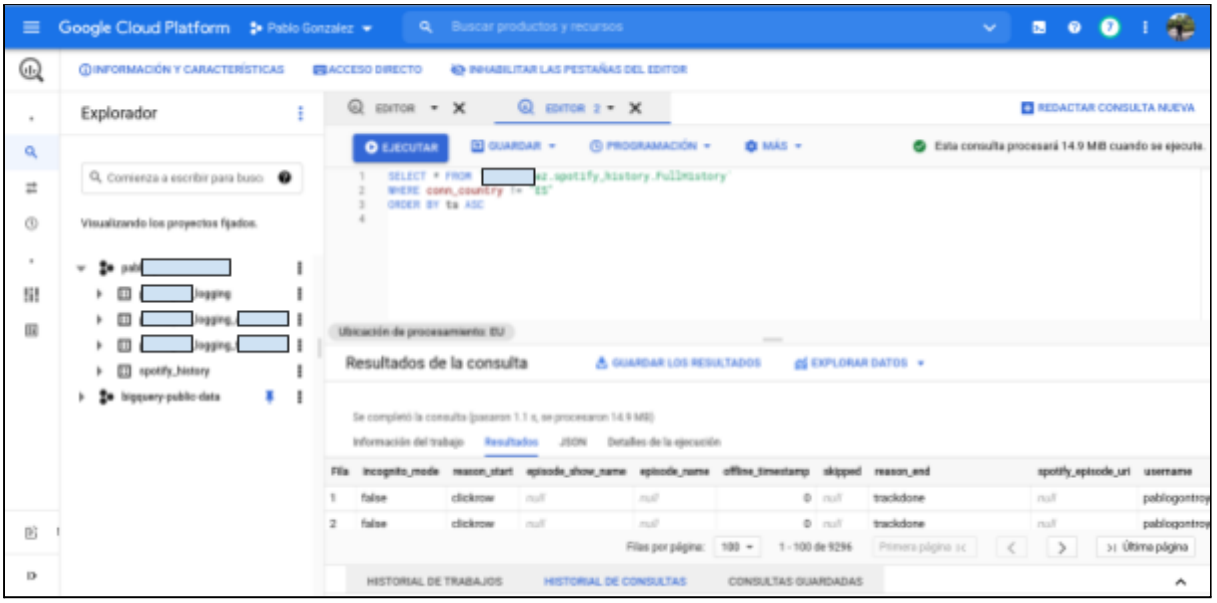
Cargar los datos

La interfaz de trabajo, accesible desde este enlace[21] nos permite realizar todas las tareas necesarias.

Primero debemos cargar los datos en un segmento de Storage, que será el origen de los datos para BigQuery. Google Cloud Platform se organiza por proyectos. En BigQuery, cada proyecto tiene conjuntos de datos. Cada conjunto de datos puede tener tablas y vistas.

La interfaz se organiza de la siguiente manera:

[20] BigQuery: almacén de datos en la nube
[21] Google Cloud Platform



Si es la primera vez que accedemos, debemos aceptar los términos. Procedemos a crear un conjunto de datos siguiendo estos pasos[22] teniendo en cuenta la ubicación y el nombre. Si no activamos la facturación en el proyecto las tablas tienen una caducidad predeterminada de 60 días, que es suficiente en la mayoría de los casos. En cualquier caso, como se ha comentado anteriormente podemos disfrutar de la capa gratuita siempre.

Siguiendo estos pasos[23] podemos crear una tabla. En cualquier caso, la configuración será similar a la siguiente. Podemos definir, y de hecho es lo más recomendable, como que BigQuery detecte el esquema de los datos. Al estar todos normalizados no tendremos problemas.

A la derecha de este texto se encuentra un ejemplo de la configuración para crear la tabla.

Una vez creada podremos consultarla, en la parte superior de esta página hay una consulta de ejemplo.

[22] cloud.google.com/bigquery/docs/quickstarts/quickstart-web-ui#create_a_dataset
[23] cloud.google.com/bigquery/docs/quickstarts/quickstart-web-ui#load_the_data_into_a_new_table

Visualización de datos e informes

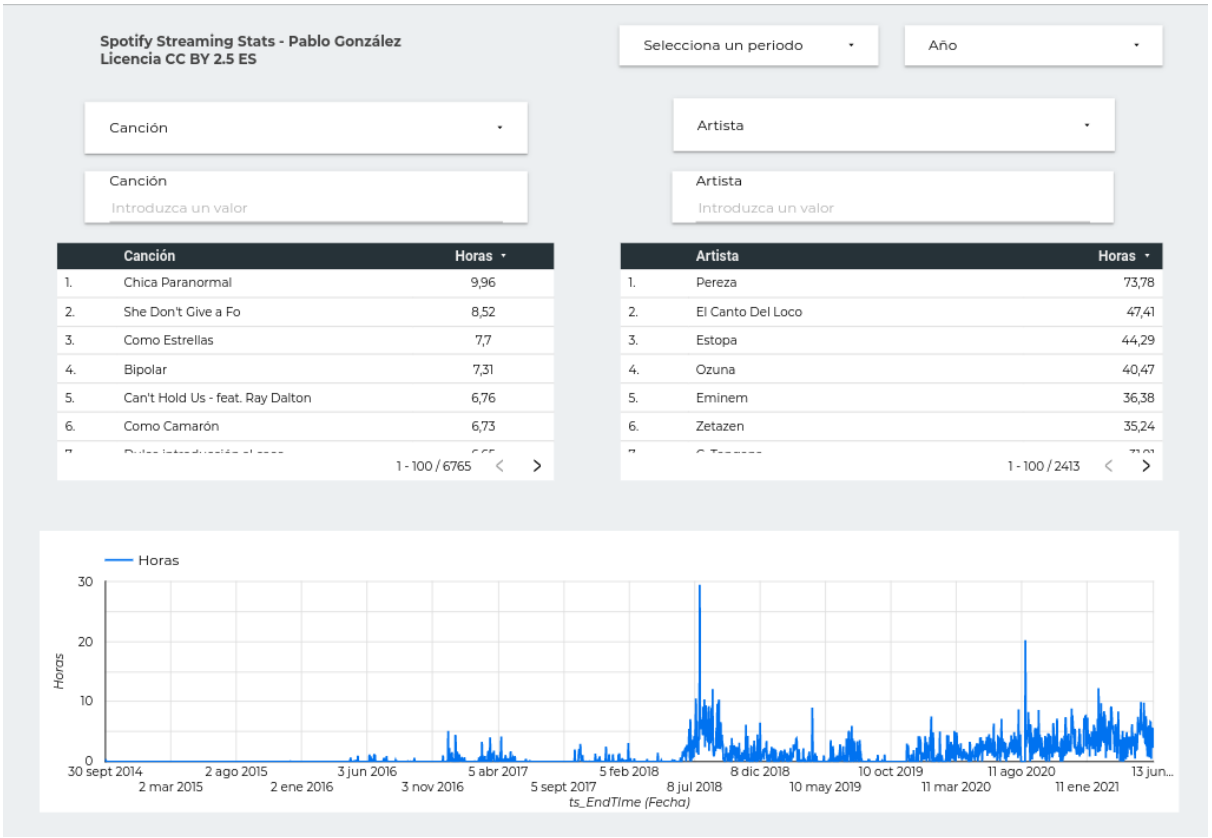
Ya tenemos los datos en nuestra base de datos, ¡genial! Pero son solo eso, datos almacenados. 70,324 entradas (filas, tuplas) y 21 columnas.

Pero hoy en día los datos sobran, somos "máquinas" generadoras de datos. Lo importante con los datos no es generarlos, sino almacenarlos, normalizarlos, presentarlos y obtener conclusiones.

Para ayudarnos a la visualización se ha elegido Google Data Studio. Esta herramienta es comparable a Microsoft Power BI o Tableau. Lo bueno es que tenemos integración total con BigQuery y la gestión de permisos con las cuentas de Google.

En esta página[24] se puede obtener información general sobre Data Studio y la documentación del producto se encuentra en este enlace[25].

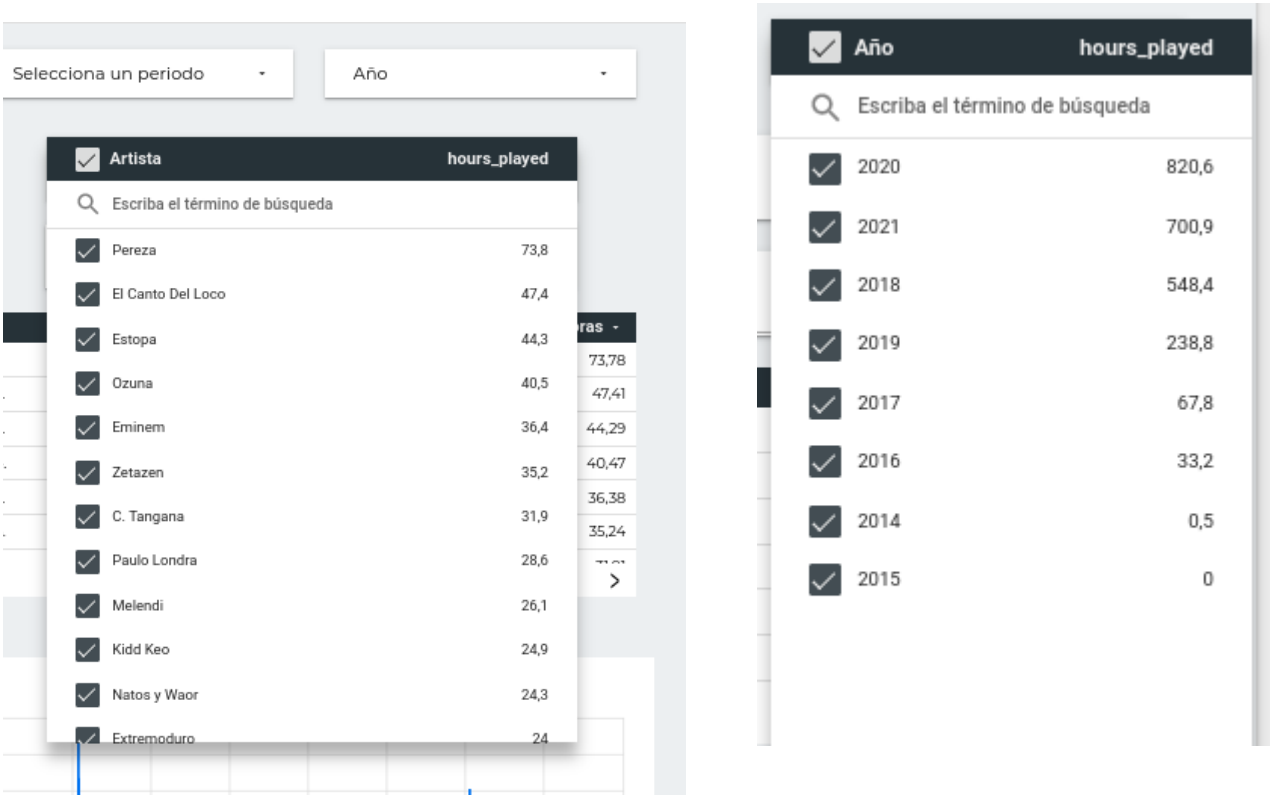
DS: Página principal y resumen



[24] Google Data Studio Overview
[25] Centro de Ayuda de Data Studio

En esta primera página vemos de un vistazo cómo han aumentado las reproducción a partir de 2018. De hecho, los años anteriores son prácticamente despreciables. Vemos a simple vista las canciones y los artistas más escuchados. Realmente fue una decepción ver las dos primeras canciones. En mi defensa se puede decir que ya no son escuchadas esas canciones y que la cuenta era compartida. En cualquier caso, es interesante.

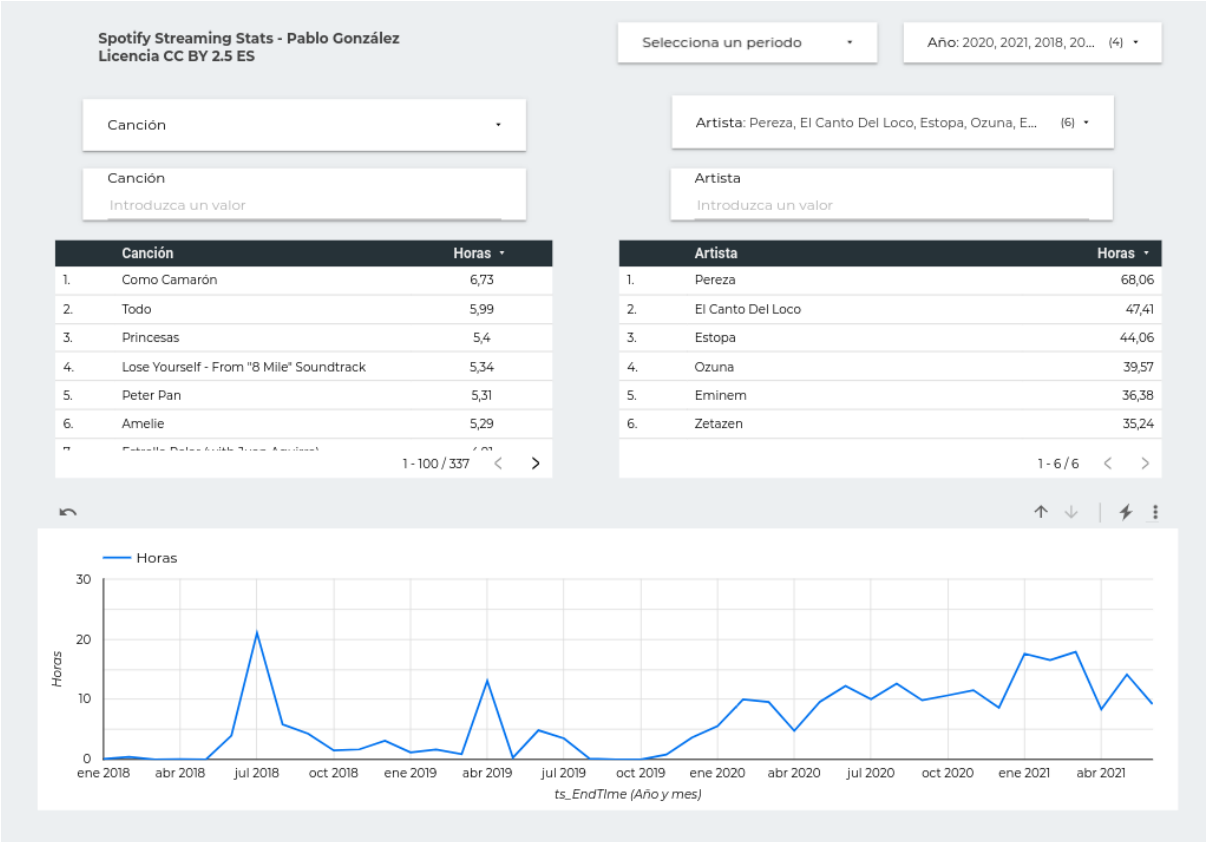
Tenemos las opciones de filtrar por canción y artista. Tenemos dos formas, escribiendo directamente o utilizando la lista desplegable. En tanto al periodo de los datos, podemos seleccionarlo por año o por días utilizando los filtros de fecha creados para ello.



En tanto al gráfico inferior, podemos agrupar por mes y año, en vez de por días. Hacer esto nos ofrecerá datos mucho más normalizados y extrapolables.

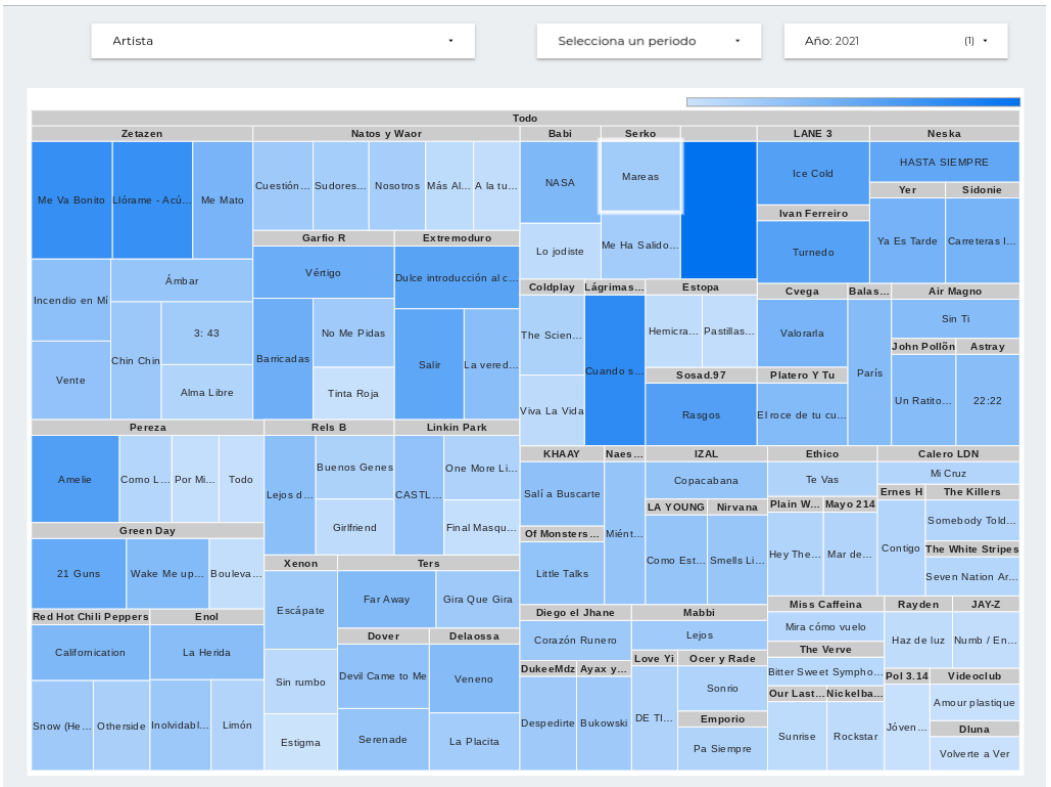
En la siguiente imagen podemos los siguientes filtros aplicados:

- Años 2018 - 2021
- Solo se muestran los 6 primeros artistas en los gráficos y tablas. Es completamente dinámico. De forma que si filtramos por un artista en las canciones más escuchadas solo nos aparecerán canciones suyas. Y el gráfico de evolución de reproducción sólo mostrará los de dicho artista.
- En el gráfico inferior los datos se muestran normalizados y agrupados por mes y año.

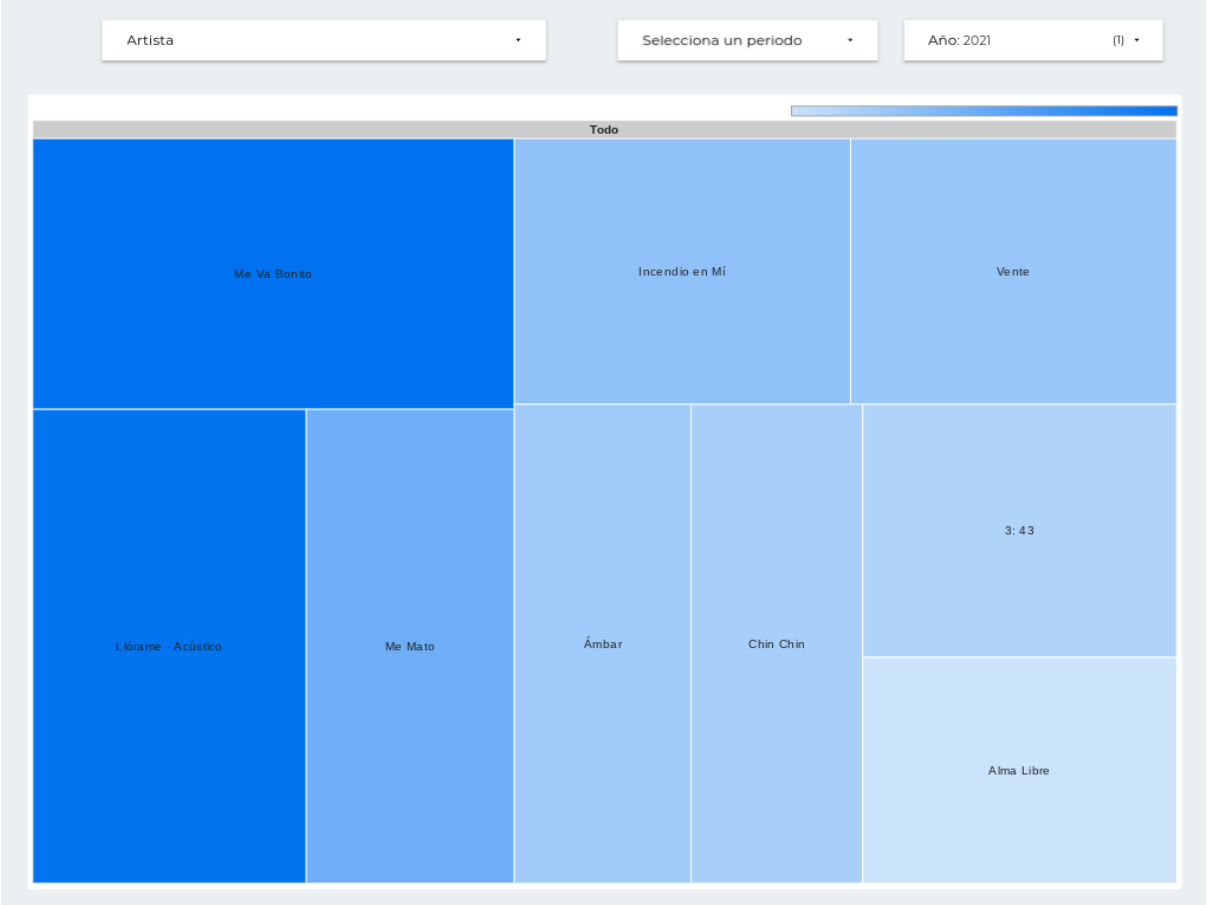


DS: Gráfico de rectángulos, simple

En la siguiente página del informe veremos las canciones más escuchadas agrupadas por artista.



Podemos, en el mismo gráfico, desglosar la información por artista. Los filtros de periodo se pueden aplicar también a esta pagina y los situados en la parte superior derecha están a nivel de informe (esto significa que se aplican aún cambiando de página). Ejemplo de desglose:



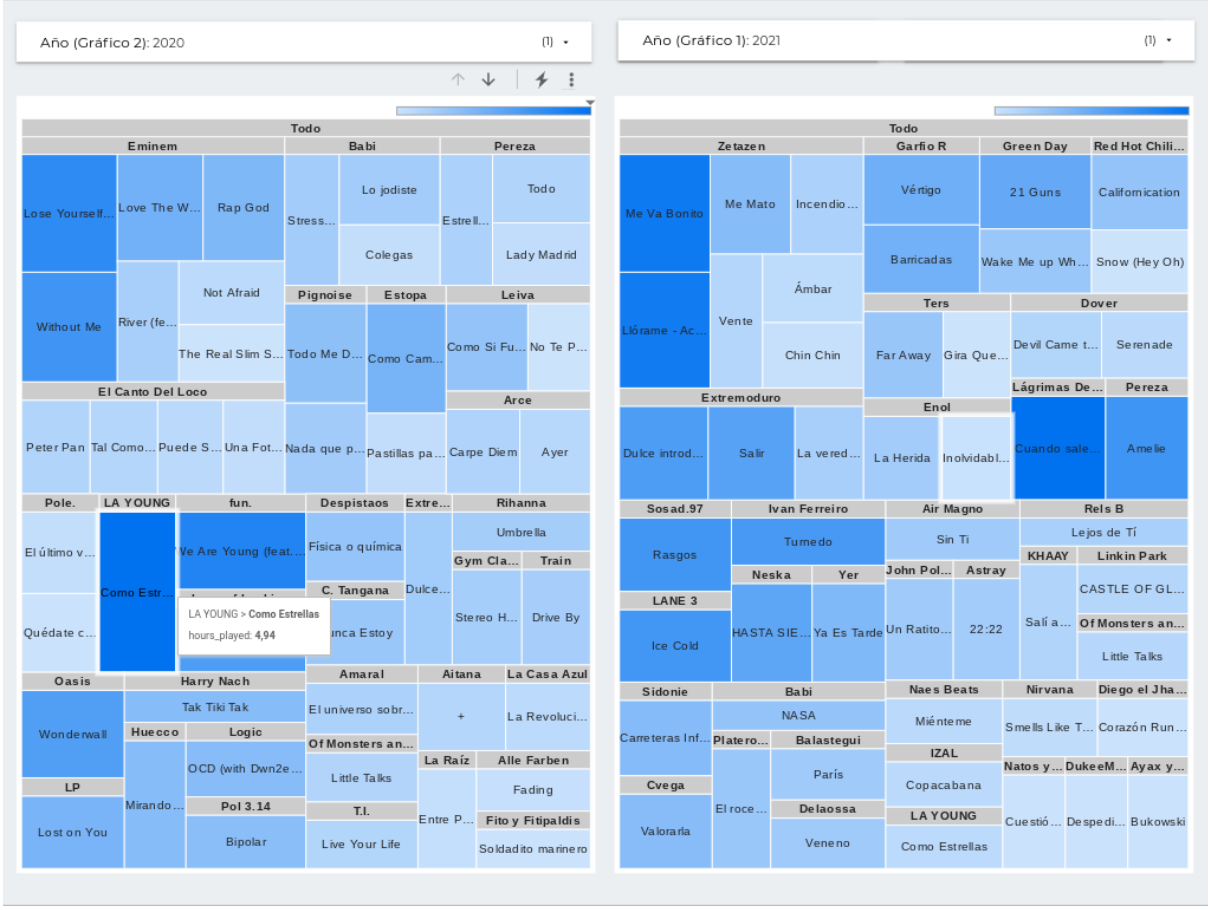
DS: Gráfico de rectángulos, comparativa

Con esta otra página tenemos la capacidad de comparar dos periodos (por defecto se muestra año en curso y el anterior) personalizados. También en gráficos de rectángulos, que permiten ver elementos destacados en base a la tonalidad del color.

Realmente nos permite ver las canciones y artistas que más se han escuchado y ver como el gusto musical evoluciona.

Al igual que en la página anterior, podemos desglosar la información mostrada por artista y aplicar filtros adicionales sobre los datos. Haciendo clic derecho sobre los gráficos podemos exportar a CSV los datos y explorarlos en una vista de Data Studio.

En la imagen a continuación se puede ver un ejemplo.



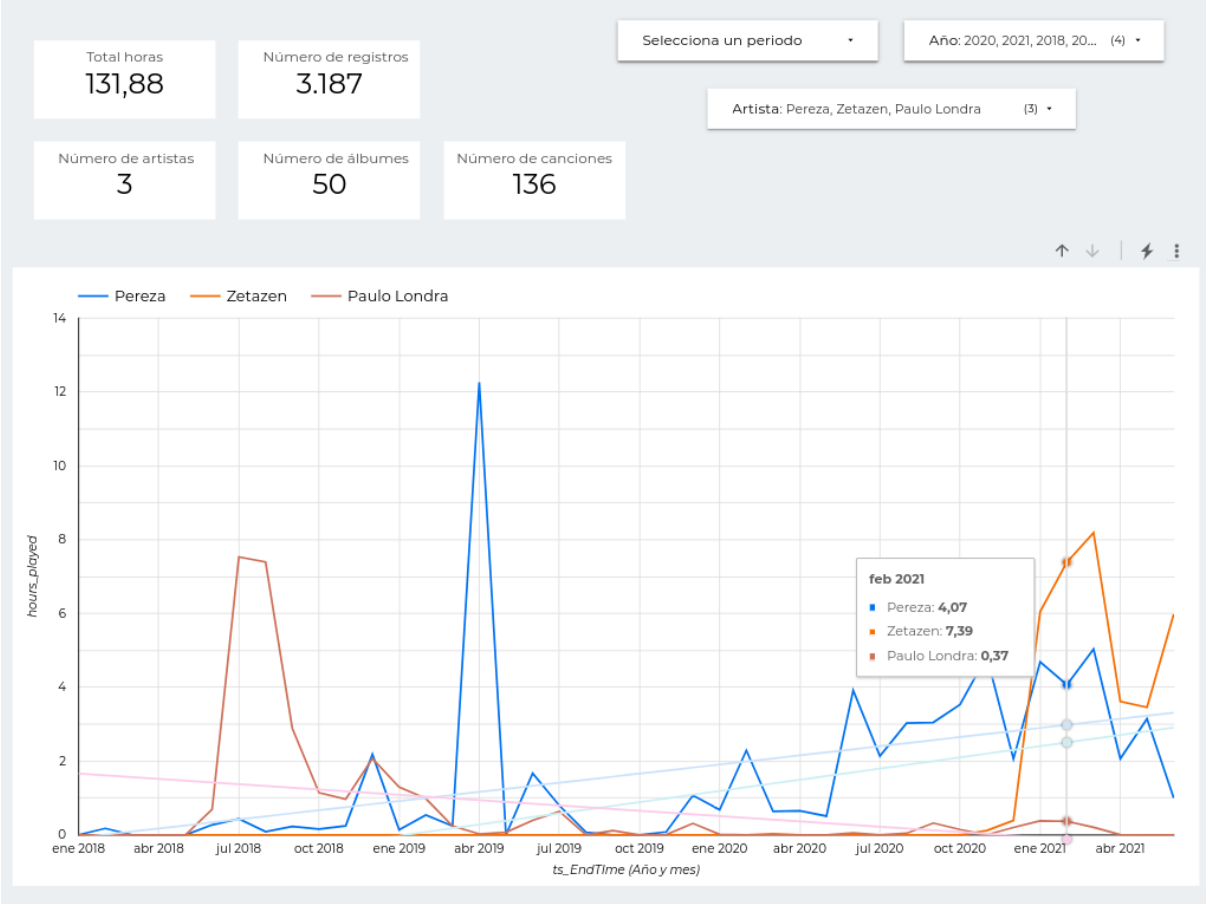
DS: Visión de tendencias

Quizá lo más importante de los datos es detectar tendencias. Los datos absolutos y las medias pueden ser inútiles[26] y desviar la atención a observaciones poco relevantes.

Del último gráfico podemos obtener una serie de conclusiones:

- Escuchar a Pereza se mantiene constante a lo largo del tiempo. Sí es cierto que se detecta un pico en abril de 2019. En el último año aumenta la media de horas escuchadas.
- En lo que respecta al artista Zetazen, si bien se ha escuchado de forma ocasional a partir del año 2021 se detecta un crecimiento muy importante. De hecho, este artista es uno de los artistas más escuchados de todo el histórico.
- Sobre el artista Paulo Londra, su representación en los últimos meses es mínima, en comparación con los datos de abril-noviembre de 2020.

[26] Si tienes 2 ojos en la cara, tienes más ojos que la media. ¿Este dato es relevante? Realmente no. Por eso es importante saber discernir entre datos.



Acceder al informe

El informe interactivo de Data Studio se encuentra disponible en este enlace[27]. Si le gustase acceder a una plantilla utilice la dirección de correo electrónico situada en la primera página de este documento.

[27] Spotify Streaming Stats - Google Data Studio