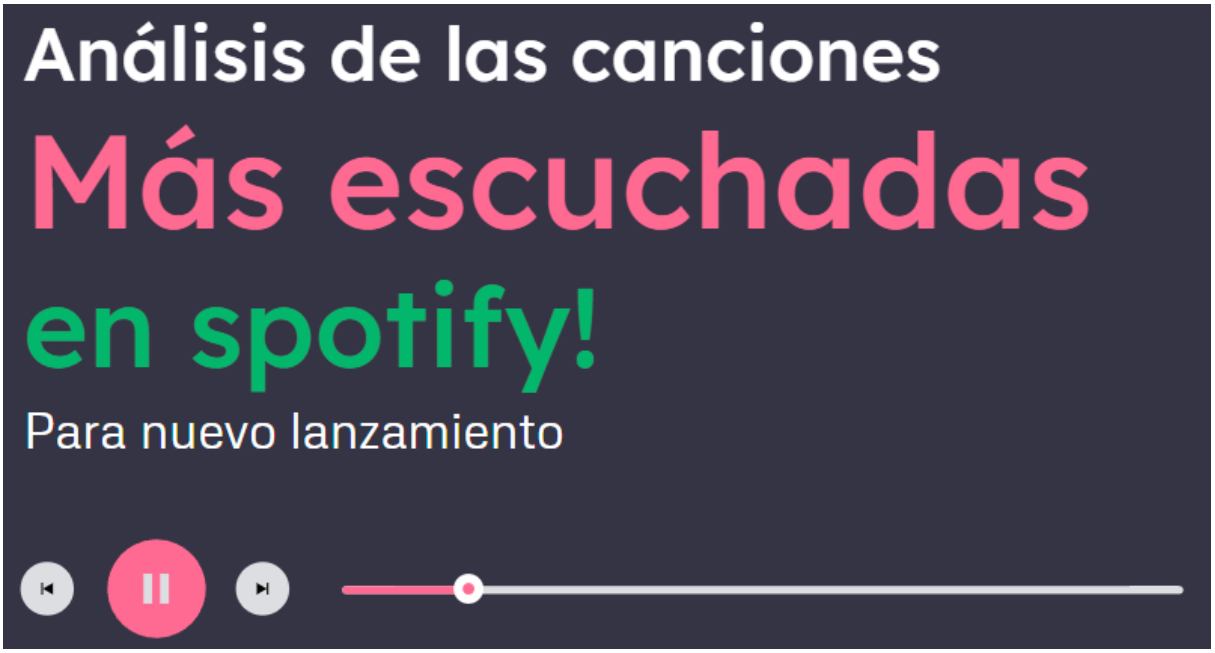
Ficha Técnica Proyecto 2: Hipótesis



- **Objetivo:**

  En este proyecto realizarás un análisis de validación de hipótesis de una base de datos de Spotify.

- **Equipo:**

  Grupal
  Sixnalie
  María José

- **Herramientas y Tecnologías:**

  BigQuery
  PowerBi

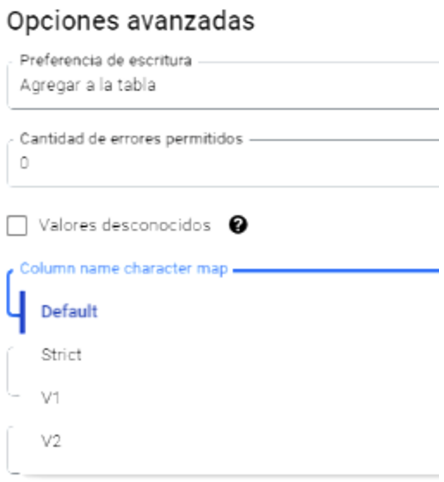
Hipótesis de cliente:

La discográfica planteó una serie de hipótesis sobre qué hace que una canción sea más escuchada. Estas hipótesis incluyen:

- Las canciones con un mayor BPM (Beats Por Minuto) tienen más éxito en términos de cantidad de streams en Spotify.
- Las canciones más populares en el ranking de Spotify también tienen un comportamiento similar en otras plataformas como Deezer.
- La presencia de una canción en un mayor número de playlists se relaciona con un mayor número de streams.
- Los artistas con un mayor número de canciones en Spotify tienen más streams.
- Las características de la canción influyen en el éxito en términos de cantidad de streams en Spotify.

Procesar y preparar base de datos:

1. **Conectar/importar datos a otras herramientas.**
   a. Descargar y descomprimir data set.
   b. Subir archivo a BigQuery:
      i. Schema/Esquema : Detección automática actividad, detecta los datos que se encuentra en cada columna, para el formato CSV.
      ii. En la tabla "**track_in_spotify"** fue necesario acudir a opciones avanzadas cambiar en la opción "Column name character map" por "V2".



      iii. Cuando se importan los datos, la columnas están con el formato "**Default"** cuando está en este modo los nombres de columnas pueden contener letras, números y algunos caracteres especiales, pero otros caracteres pueden ser restringidos o convertidos a un formato estándar, por eso **"track_in_spotify"** presentaba error al ser importado.

      Ponerlo con el formato "
      V2" con este es el modo más reciente puede incluir mejoras o cambios en cómo se manejan ciertos caracteres en los nombres de columnas.

2. **Identificar y manejar valores nulos.**

a) En la tabla **"track_in_spotify"** no se encuentran datos nulos. Para verificarlo, usamos el código en el SQL:

```
SELECT
  *
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify`
WHERE
  track_id IS NULL
  OR track_name IS NULL
  OR artist_s__name IS NULL
  OR artist_count IS NULL
  OR released_year IS NULL
```

```
  OR released_month IS NULL
  OR released_day IS NULL
  OR in_spotify_playlists IS NULL
  OR in_spotify_charts IS NULL
  OR streams IS NULL
```

Se utiliza el "*" después de "SELECT" para que pueda verificar en todas las columnas la información de la tabla.

b) En la tabla **"track_in_competition"** se encontraron 50 datos nulos en la columna "in_shazam_charts". Para verificarlo, usamos el código en el SQL:

```
SELECT
*
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_competition`
WHERE
  track_id IS NULL
  OR in_apple_playlists IS NULL
  OR in_apple_charts IS NULL
  OR in_deezer_playlists IS NULL
  OR in_deezer_charts IS NULL
  OR in_shazam_charts IS NULL
```

https://docs.google.com/spreadsheets/d/1vz6GLlK7BsZFq7KcJV5y__kFKmg__DEonkgjLVr-BMM/edit?usp=sharing

c) En la tabla **"track_technical"** se encontraron 95 datos nulos en la columna "key". Aquí tuvimos que agregarle **'** a las columnas de tuviesen en su nombre un carácter especial (% en este caso) para que el sql pudiera seguir la orden de verificar si hay datos nulos. Usamos el siguiente código:

```
SELECT
  *
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_technical_info`
WHERE
  track_id IS NULL
  OR bpm IS NULL
  OR KEY IS NULL
  OR mode IS NULL
  OR `danceability_%` IS NULL
  OR `valence_%` IS NULL
  OR `energy_%` IS NULL
  OR `acousticness_%` IS NULL
  OR `instrumentalness_%` IS NULL
  OR `liveness_%` IS NULL
  OR `speechiness_%` IS NULL;
```

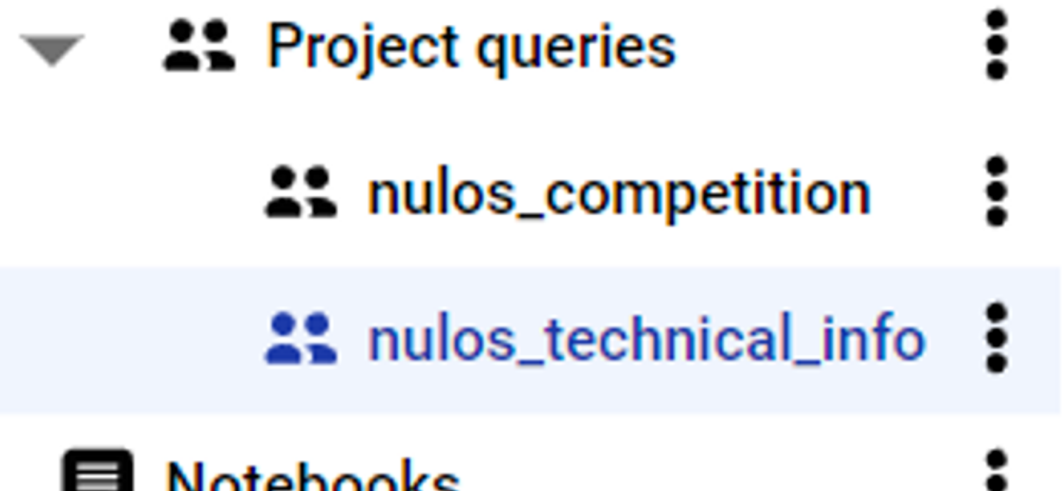
https://docs.google.com/spreadsheets/d/11dWhBRYtwLRNkoldxEkvEv_F40JMNmjZAw_eASMGBNg/edit?usp=sharing

e) Buscamos qué canciones son las que aparecen como *null* en la columna de "in_shazam_charts". Son 50 canciones que ahora sabemos a qué artistas pertenecen. Quizás puede ser un dato irrelevante pero

se aprendió a realizar este tipo de búsqueda.

```
SELECT
  tc.track_id,
  ts.track_name,
  ts.artist_s__name,
  ts.artist_count,
  ts.released_year,
  ts.released_month,
  ts.released_day,
  ts.in_spotify_playlists,
  ts.in_spotify_charts,
  ts.streams,
  tc.in_apple_playlists,
  tc.in_apple_charts,
  tc.in_deezer_playlists,
  tc.in_deezer_charts,
  tc.in_shazam_charts
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_competition` AS tc
JOIN
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify` AS ts
ON
  tc.track_id = ts.track_id
WHERE
  tc.in_shazam_charts IS NULL;
```

d) Las consultas (query) fueron guardadas para llegar a un consenso con su uso



3. **Identificar y manejar valores duplicados**

a) En la tabla **“track_in_spotify”** se encuentran estos duplicados

Row	track_name	artist_s__name	cantidad
1	SNAP	Rosa Linn	2
2	About Damn Time	Lizzo	2
3	Take My Breath	The Weeknd	2
4	SPIT IN MY FACE!	ThxSoMch	2

El resultado será una lista de nombres de pistas (`track_name`) y nombres de artistas (`artist_s__name`) que aparecen más de una vez en la tabla, junto con la cantidad de veces que aparecen (`cantidad`)

```
SELECT
  track_name,
  artist_s__name,
  COUNT(*) AS cantidad
FROM
  `mjproyecto2.database.track_in_spotify`
GROUP BY track_name,artist_s__name
HAVING COUNT(*) >1
```

```
HAVING COUNT(*) > 1
```

Esta cláusula filtra los grupos creados por la cláusula `GROUP BY` . En este caso, solo se incluirán en los resultados aquellos grupos donde el número de registros (el conteo) es mayor que 1. En otras palabras, esta cláusula selecciona solo los `track_name` y `artist_s__name` que aparecen más de una vez en la tabla, lo cual indica duplicados.

- Decidimos investigar en detalle los datos duplicados para poder tomar una decisión informada sobre su manejo. Para ello, realizamos una consulta que nos permitió examinar minuciosamente los detalles de la tabla completa. Como conclusión, optamos por conservar los registros que presentaran las cifras más altas en las columnas "in_spotify_playlist", "in_spotify_charts" y "streams".

```
SELECT
  t1.track_name,
  t1.artist_s__name,
  t1.track_id,
  t1.artist_count,
  t1.released_year,
  t1.released_month,
  t1.released_day,
  t1.in_spotify_playlists,
  t1.in_spotify_charts,
  t1.streams,
  t2.cantidad
FROM
  `mjproyecto2.database.track_in_spotify` t1
```

```
JOIN (
  SELECT
    track_name,
    artist_s__name,
    COUNT(*) AS cantidad
  FROM
    `mjproyecto2.database.track_in_spotify`
  GROUP BY
    track_name,
    artist_s__name
  HAVING
    COUNT(*) > 1 ) t2
ON
  t1.track_name = t2.track_name
  AND t1.artist_s__name = t2.artist_s__name
ORDER BY
  t1.track_name,
  t1.artist_s__name;
```

Row	track_name	artist_s__name	track_id	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams
1	About Damn Time	Lizzo	7173596	1	2022	4	14	9021	0	723894473
2	About Damn Time	Lizzo	5080031	1	2022	7	15	2332	2	723894473
3	SNAP	Rosa Linn	5675634	1	2022	3	19	3202	18	726307468
4	SNAP	Rosa Linn	3814670	1	2022	3	19	1818	0	711366595
5	SPIT IN MY FACE!	ThxSoMch	4967469	1	2022	10	31	629	14	303216294
6	SPIT IN MY FACE!	ThxSoMch	8173823	1	2022	10	31	573	0	301869854
7	Take My Breath	The Weeknd	1119309	1	2021	8	6	2597	0	130655803
8	Take My Breath	The Weeknd	4586215	1	2021	8	6	6392	0	432702334

b) En la tabla **“track_in_competition”** solo decidimos buscar duplicados en **“track_id”**, ya que en las demás columnas no son de relevancia los datos duplicados

```
SELECT
  track_id, COUNT(*) AS cantidad
FROM
  `mjproyecto2.database.track_in_competition`
GROUP BY
  track_id
HAVING
  COUNT(*) >1
```

There is no data to display.

c) En la tabla **“track_technical_info”** solo decidimos bucar duplicados en **“track_id”**, ya que en las demás columnas no son de relevancia los datos duplicados

```
SELECT
  track_id, COUNT(*) AS cantidad
FROM
  `mjproyecto2.database.track_technical_info`
GROUP BY
```

```
  track_id
HAVING
  COUNT(*) >1
```

4. **Identificar y manejar datos fuera del alcance del análisis.**

a) Descartamos de “track_in_technical” las columnas “Key” y “mode” ya que no son necesarias para la investigación.

```
SELECT
  * EXCEPT (key, mode)
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_technical_info`
```

5.
Identificar y manejar datos discrepantes en variables categóricas.

a) Algunos nombres de artistas y canciones tienen símbolos raros. Utilizamos REGEXP_REPLACE para crear una columna limpia sin esos símbolos pero con los nombres originales.

```
SELECT
  track_name,
  REGEXP_REPLACE(track_name, r'[^a-zA-Z0-9 ]', '') AS track_name_limpio,
  artist_s__name,
  REGEXP_REPLACE(artist_s__name, r'[^a-zA-Z0-9 ]', '') AS artist_s__name_limpi
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify`
```

6. **Identificar y manejar datos discrepantes en variables numéricas.**

a) En la tabla **“track_in_spotify”** se encuentra:

Row	track_id	streams
1	4061483	BPM110KeyAModeMajorDanceability53Valence75Energy69Acousticness7Instrumentalness0Liveness17Speechiness3

```
SELECT
  track_id,
  streams
FROM
```

```
  `mjproyecto2.database.track_in_spotify`
WHERE
  SAFE_CAST(streams AS INT64) IS NULL
```

Para seleccionar las filas en las que el campo `streams` contiene datos no numéricos en BigQuery, puedes utilizar la función `SAFE_CAST` para intentar convertir los valores de `streams` a números y luego filtrar aquellos valores que no se pueden convertir.

Uso de `INT64` en BigQuery

En BigQuery, `INT64` es un tipo de dato que representa números enteros de 64 bits. Es adecuado para trabajar con números enteros grandes que no tienen decimales.

Si queremos que no se refleje ese dato en la tabla, aplicamos la siguiente fórmula:

```
SELECT
  tc.track_id,
  ts.track_name,
  ts.artist_s__name,
  ts.artist_count,
  ts.released_year,
  ts.released_month,
  ts.released_day,
  ts.in_spotify_playlists,
  ts.in_spotify_charts,
  ts.streams,
  tc.in_apple_playlists,
  tc.in_apple_charts,
  tc.in_deezer_playlists,
  tc.in_deezer_charts,
  tc.in_shazam_charts
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_competition` AS tc
JOIN
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify` AS ts
ON
  tc.track_id = ts.track_id
WHERE
  SAFE_CAST(ts.streams AS INT64) IS NOT NULL;
```

Hay que tomar en cuenta que están 2 tablas, la de “track_in_competition” y “track_in_spotify”.

En la tabla "track_in_spotify", decidimos identificar los registros duplicados y, para el análisis, conservar solo las canciones con los mayores valores. Para ello, creamos una consulta que nos permite separar los duplicados con los menores valores.

Row	track_name	artist_s__name	track_id	artist	releasec	releas	release	in_spotify_pla	in_spot	streams
1	About Damn Time	Lizzo	5080031	1	2022	7	15	2332	2	723894473
2	SNAP	Rosa Linn	3814670	1	2022	3	19	1818	0	711366595
3	SPIT IN MY FACE!	ThxSoMch	8173823	1	2022	10	31	573	0	301869854
4	Take My Breath	The Weeknd	1119309	1	2021	8	6	2597	0	130655803

```
WITH RankedTracks AS (
  SELECT
    t1.track_name,
    t1.artist_s__name,
    t1.track_id,
    t1.artist_count,
    t1.released_year,
    t1.released_month,
    t1.released_day,
    t1.in_spotify_playlists,
    t1.in_spotify_charts,
    t1.streams,
    ROW_NUMBER() OVER (
      PARTITION BY t1.track_name, t1.artist_s__name
      ORDER BY t1.in_spotify_playlists ASC, t1.streams ASC
    ) AS rank
  FROM
    `mjproyecto2.database.track_in_spotify` t1
  JOIN (
    SELECT
      track_name,
      artist_s__name,
      COUNT(*) AS cantidad
    FROM
      `mjproyecto2.database.track_in_spotify`
    GROUP BY
      track_name,
      artist_s__name
    HAVING
      COUNT(*) > 1
  ) t2
  ON
    t1.track_name = t2.track_name
    AND t1.artist_s__name = t2.artist_s__name
)
SELECT
  track_name,
  artist_s__name,
  track_id,
  artist_count,
  released_year,
  released_month,
  released_day,
```

```
  in_spotify_playlists,
  in_spotify_charts,
  streams
FROM
  RankedTracks
WHERE
  rank = 1
ORDER BY
  track_name,
  artist_s__name;
```

7. **Comprobar y cambiar tipo de dato.**

a) Modificando el tipo de dato de la columna “streams”, encontramos que hay variables numéricas que hacen que falle la modificación.

```
SELECT
  CAST (streams AS int64) AS streams_limpio
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify`
WHERE
  SAFE_CAST(streams AS int64) IS NOT NULL
```

con este código podemos descartar los datos problemáticos (total de datos problemáticos: 1).

Ahora, si queremos crear una nueva tabla donde la columna “streams” deje de ser STRING y sea completamente INT64 o INTEGER, debemos:

- Seleccionar el proyecto y conjunto de datos.
- Abrir una nueva pestaña de consultas.
- Copiar y pegar la siguiente consulta:

```
CREATE TABLE `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify_modificad
SELECT
  track_id,
  track_name,
  artist_s__name,
  artist_count,
  released_year,
  released_month,
  released_day,
  in_spotify_playlists,
  in_spotify_charts,
  SAFE_CAST(streams AS INT64) AS streams
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify`
```

```
WHERE
  SAFE_CAST(streams AS INT64) IS NOT NULL;
```

- Ejecutar la consulta.

Así la nueva tabla tendrá la información que teníamos anteriormente (sin el dato problemático) y con el tipo de dato de la columna “streams” cambiado. Y la nueva hoja para trabajar se llama **‘track_in_spotify_modificado’.**

Ahora para tener un mejor orden con el proyecto, podemos crear la nueva tabla de spotify con los datos limpios:

```
CREATE TABLE
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify_modificado` AS
SELECT
  track_id,
  REGEXP_REPLACE(track_name, r'[^a-zA-Z0-9 ]', '') AS track_name_limpio,
  REGEXP_REPLACE(artist_s__name, r'[^a-zA-Z0-9 ]', '') AS artists_name_limpio,
  artist_count,
  released_year,
  released_month,
  released_day,
  in_spotify_playlists,
  in_spotify_charts,
  SAFE_CAST(streams AS INT64) AS streams
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify`
WHERE
  SAFE_CAST(streams AS INT64) IS NOT NULL
  AND track_id NOT IN ('5080031',
    '8173823',
    '1119309',
    '3814670');
```

Con este tendremos una nueva tabla de datos con:

- Dato cambiado en la columna “streams”.
- Sin el dato problemático que se encontraba en la columna streams.
- Sin duplicados con menor valor (criterio de nosotras).
- Titular arreglado en las columnas track_name y artists_name.

8. **Crear nuevas variables**

a) Para crear la variable “fecha_de_lanzamiento” utilizamos CONCAT y CAST para que la fecha quede en el formato aaaa-mm-dd:

```
SELECT
CONCAT(
  CAST(released_year AS STRING), '-',
  RIGHT('0' || CAST(released_month AS STRING), 2), '-',
```

```
  RIGHT('0' || CAST(released_day AS STRING), 2)
  ) AS fecha_de_lanzamiento
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify_modificado`
```

Explicación de códigos usados en la fórmula:

- **SELECT**: Esta cláusula se utiliza para especificar las columnas que deseas recuperar de la tabla.
- **CONCAT()**: La función `CONCAT` se utiliza para concatenar (unir) varias cadenas en una sola cadena.
- **CAST(released_year AS STRING)**: La función `CAST` convierte el valor de `released_year` a una cadena de texto (`STRING`). Esto es necesario porque `CONCAT` funciona con cadenas, no con números.
- **'-'**: Es una cadena literal que representa el guion que separa los componentes del año, mes y día en el formato de fecha `aaaa-mm-dd`.
- **RIGHT('0' || CAST(released_month AS STRING), 2)**:
  - `CAST(released_month AS STRING)`: Convierte el valor de `released_month` a una cadena.
  - `'0' || CAST(released_month AS STRING)`: Concatenación del carácter '0' al valor del mes convertido en cadena. Esto asegura que los meses de un solo dígito (por ejemplo, `3` para marzo) se conviertan en dos dígitos (`03`).
  - `RIGHT(..., 2)`: La función `RIGHT` toma los dos caracteres más a la derecha de la cadena resultante. Así, si el mes era `3`, la cadena `'03'` se convierte en `03`.
- **RIGHT('0' || CAST(released_day AS STRING), 2)**:
  - Similar a lo anterior, esta parte del comando asegura que los días de un solo dígito (por ejemplo, `7` para el séptimo día del mes) se conviertan en dos dígitos (`07`).
- **AS fecha_de_lanzamiento**: Esta cláusula asigna un alias a la columna resultante. Este es el nombre que tendrá la columna de fecha en el resultado: *"fecha_de_lanzamiento"* .
- **FROM (nombre de la tabla)**: Esta cláusula especifica la tabla de la cual se están seleccionando los datos. En este caso: *`proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify_modificado`* .

En esta fórmula no hemos modificado el tipo de dato. En este caso queremos que esta columna se maneje con el dato "DATE" que se utiliza para representar fechas (año, mes, día).

```
SELECT
CAST (
  CONCAT(
    CAST(released_year AS STRING), '-',
    RIGHT('0' || CAST(released_month AS STRING), 2), '-',
    RIGHT('0' || CAST(released_day AS STRING), 2)
    ) AS DATE
)AS fecha_de_lanzamiento
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify_modificado`
```

La razón por la que se utilizan datos de tipo `STRING` en lugar de `DATE` es porque estamos manipulando y concatenando las partes de la fecha como cadenas de texto antes de convertirlas a un formato de fecha (`DATE`). Aquí está la explicación detallada:

- **Concatenación y Formateo**: Crea la cadena de texto en el formato `aaaa-mm-dd` utilizando las funciones `CAST`, `RIGHT` y `CONCAT` como se explicó anteriormente.
- **Conversión a DATE**: Utiliza `CAST` nuevamente, esta vez para convertir la cadena de texto resultante en un tipo de dato `DATE`.

Al final, *"fecha_de_lanzamiento"* será de tipo `DATE` en lugar de `STRING`.

Resumen de Pasos:

1. Convertir las partes de la fecha a cadenas (`CAST` a `STRING`).
2. Asegurar que el mes y el día tengan dos dígitos (`RIGHT`).
3. Concatenar las partes en una cadena en el formato `aaaa-mm-dd` (`CONCAT`).
4. (Opcional) Convertir la cadena formateada a tipo `DATE` (`CAST` a `DATE`).

b) Para crear la variable *"total_en_playlists"* creamos una nueva tabla de "track_in_competition_modificado" para no reflejar los datos duplicados y el dato problemático que encontramos anteriormente, para esto se realiza este procedimiento:

- Seleccionar el proyecto y conjunto de datos.
- Abrir una nueva pestaña de consultas.
- Copiar y pegar la siguiente consulta:

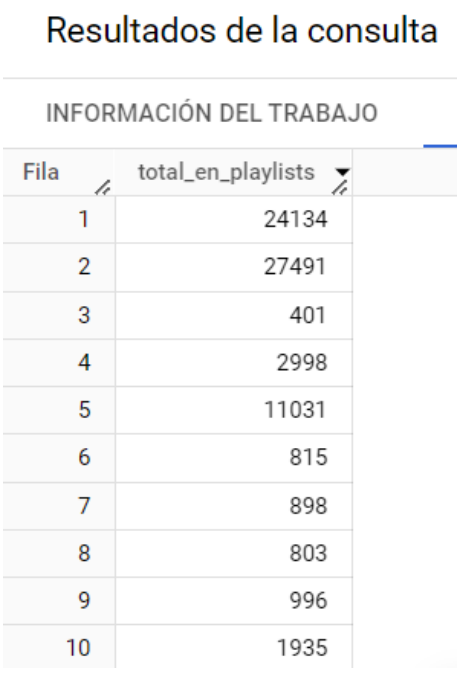
```
CREATE TABLE
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_competition_modificado` AS
SELECT
  track_id,
  in_apple_playlists,
  in_apple_charts,
  in_deezer_playlists,
  in_deezer_charts,
  in_shazam_charts

FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_competition`
WHERE
 track_id NOT IN ('5080031',
    '8173823',
    '1119309',
    '3814670',
    '4061483');
```

- Ejecutar la consulta.

Una vez creada la nueva tabla (sin los duplicados y sin el dato problemático) se procede a crear la nueva variable:

```
SELECT
  (IFNULL(t1.in_apple_playlists, 0) +
  IFNULL(t1.in_deezer_playlists, 0) +
  IFNULL(t2.in_spotify_playlists, 0)) AS total_en_playlists
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_competition_modificado` t1
LEFT JOIN
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify_modificado` t2
ON
  t1.track_id = t2.track_id;
```

Las filas que tenían datos “0” los consideraba la tabla como dato *null*. Entonces para que contara esos datos en el cálculo de `total_de_playlists`, utilizamos la función `IFNULL` para que pudiera contarlos en la lista.



Resultados de la consulta

INFORMACIÓN DEL TRABAJO

Fila	total_en_playlists
1	24134
2	27491
3	401
4	2998
5	11031
6	815
7	898
8	803
9	996
10	1935

9. **Unir tablas**

a) Antes de unir tablas, procedimos a crear view (vista) con los datos limpios de cada tabla

- View para track_spotify

```
CREATE VIEW `mjproyecto2.database.view_ts` AS
SELECT
  track_id,
  REGEXP_REPLACE(track_name, r'[^a-zA-Z0-9 ]', '') AS track_name_limpio,
```

```
  REGEXP_REPLACE(artist_s__name, r'[^a-zA-Z0-9 ]', '') AS artist_s__name_limpi
  artist_count,
  released_year,
  released_month,
  released_day,
  in_spotify_playlists,
  in_spotify_charts,
  SAFE_CAST(streams AS INT64) AS streams,
  CONCAT(
    CAST(released_year AS STRING), '-',
    LPAD(CAST(released_month AS STRING), 2, '0'), '-',
    LPAD(CAST(released_day AS STRING), 2, '0')
  ) AS fecha_completa
FROM
  `mjproyecto2.database.track_in_spotify`
WHERE
  SAFE_CAST(streams AS INT64) IS NOT NULL
  AND track_id NOT IN ('5080031', '8173823', '1119309', '3814670', '0:00');
```

- View para track_competition

```
CREATE VIEW `proyecto-2-hipotesis-426800.spotify_2023.view_tc` AS
SELECT
  tc.track_id,
  tc.in_apple_playlists,
  tc.in_apple_charts,
  tc.in_deezer_playlists,
  tc.in_deezer_charts,
  tc.in_shazam_charts,
  ts.in_spotify_playlists,
  (IFNULL(tc.in_apple_playlists, 0) + IFNULL(tc.in_deezer_playlists, 0) + IFNUL
  (IFNULL(tc.in_apple_charts, 0) + IFNULL(tc.in_deezer_charts, 0) + IFNULL(tc.
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_competition` tc
LEFT JOIN
  `proyecto-2-hipotesis-426800.spotify_2023.track_in_spotify` ts
ON
  tc.track_id = ts.track_id
WHERE
  tc.track_id NOT IN ('5080031', '8173823', '1119309', '3814670', '4061483', '
```

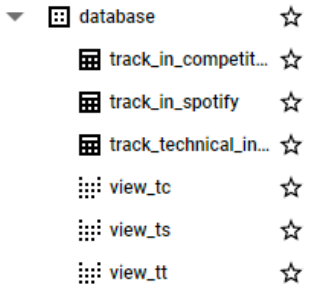
Explicación:

1. `tc` es un alias para `track_in_competition`.
2. `ts` es un alias para `track_in_spotify`.
3. Se utiliza un `LEFT JOIN` para combinar las tablas `track_in_competition` y `track_in_spotify` basándose en el campo `track_id`.
4. Se agrega el campo `in_spotify_playlists` de la tabla `track_in_spotify` al SELECT.
5. La suma para `total_en_playlist` ahora incluye `IFNULL(tsp.in_spotify_playlists, 0)`.

Esto te permitirá obtener el total de playlists incluyendo la información de Spotify.

- View para track_technical

```
CREATE VIEW `mjproyecto2.database.view_tt` AS
SELECT
  * EXCEPT (key, mode)
FROM
  `mjproyecto2.database.track_technical_info`
WHERE
  track_id NOT IN ('5080031', '8173823', '1119309', '3814670', '4061483', '0:0(
```



b) Obtenidas las view, unimos los datos:

```
SELECT *
FROM `database.view_ts` AS TS
LEFT JOIN `database.view_tc` AS TC
ON TS.track_id = TC.track_id
LEFT JOIN `database.view_tt` AS TT
ON TS.track_id = TT.track_id;
```

Para evitar que la columna `track_id` se repita varias veces en la tabla resultante, seleccionamos explícitamente las columnas que necesitábamos incluir en el resultado en lugar de utilizar `SELECT *`. Esto nos permite omitir las columnas duplicadas.

```
CREATE VIEW mjproyecto2.database.vista_track_data AS
SELECT
  TS.track_id,
  REGEXP_REPLACE(TS.track_name, r'[^a-zA-Z0-9 ]', '') AS track_name_limpio,
  REGEXP_REPLACE(TS.artist_s__name, r'[^a-zA-Z0-9 ]', '') AS artist_s__name_limpi
  TS.artist_count,
  TS.released_year,
  TS.released_month,
  TS.released_day,
  TS.in_spotify_playlists,
  TS.in_spotify_charts,
  SAFE_CAST(TS.streams AS INT64) AS streams,
  CONCAT(
```

```
    CAST(TS.released_year AS STRING), '-',
    LPAD(CAST(TS.released_month AS STRING), 2, '0'), '-',
    LPAD(CAST(TS.released_day AS STRING), 2, '0')
  ) AS fecha_completa,
  TC.in_apple_playlists,
  TC.in_apple_charts,
  TC.in_deezer_playlists,
  TC.in_deezer_charts,
  TC.in_shazam_charts,
  (IFNULL(TC.in_apple_playlists, 0) + IFNULL(TC.in_deezer_playlists, 0)) AS total
  (IFNULL(TC.in_apple_charts, 0) + IFNULL(TC.in_deezer_charts, 0) + IFNULL(TC.in_
  TT.bpm,
  TT.`danceability_%`,
  TT.`valence_%`,
  TT.`energy_%`,
  TT.`acousticness_%`,
  TT.`instrumentalness_%`,
  TT.`liveness_%`,
  TT.`speechiness_%`
FROM
  `mjproyecto2.database.track_in_spotify` TS
LEFT JOIN `mjproyecto2.database.track_in_competition` TC
  ON TS.track_id = TC.track_id
LEFT JOIN `mjproyecto2.database.track_technical_info` TT
  ON TS.track_id = TT.track_id
WHERE
  SAFE_CAST(TS.streams AS INT64) IS NOT NULL
  AND TS.track_id NOT IN ('5080031', '8173823', '1119309', '3814670')
  AND TC.track_id NOT IN ('5080031', '8173823', '1119309', '3814670', '4061483')
  AND TT.track_id NOT IN ('5080031', '8173823', '1119309', '3814670', '4061483');
```

Ahora, este código es muy extenso. Conseguimos otra manera de crear las tablas y que no se repitiera la columnas con un código más corto. Utilizamos la cláusula `USING` en lugar de `ON` para simplificar el código cuando las columnas que queremos unir tienen el mismo nombre. Esto evita que las columnas duplicadas aparezcan en el resultado.

```
SELECT *
FROM `proyecto-2-hipotesis-426800.spotify_2023.view_ts` AS TS
LEFT JOIN `proyecto-2-hipotesis-426800.spotify_2023.view_tc` AS TC
USING (track_id)
LEFT JOIN `proyecto-2-hipotesis-426800.spotify_2023.view_tt` AS TT
USING (track_id);
```

En este caso, la columna `track_id` se unirá utilizando `USING`, lo que garantiza que no se repetirá en el resultado final. Este sería el código que usaremos para unir los datos.

💡 OBSERVACIÓN (por otras columnas tienen nombres iguales en distintas tablas)

Para track_competition como existe una suma entre tablas para el resultado de “total_en_playlists”, se duplica el nombre “in_spotify_playlists”, entonces se plantea esta fórmula:

```
SELECT
    TS.track_id,
    COALESCE(TS.in_spotify_playlists, TC.in_spotify_playlists) AS in_spotify_play
    TS.* EXCEPT (track_id, in_spotify_playlists),
    TC.* EXCEPT (track_id, in_spotify_playlists),
    TT.* EXCEPT (track_id)
FROM
    `proyecto-2-hipotesis-426800.spotify_2023.view_ts` AS TS
LEFT JOIN
    `proyecto-2-hipotesis-426800.spotify_2023.view_tc` AS TC
USING (track_id)
LEFT JOIN
    `proyecto-2-hipotesis-426800.spotify_2023.view_tt` AS TT
USING (track_id)
```

Se usa únicamente si el nombre de las columnas es igual y existe problema de duplicados.

10. **Construir tablas auxiliares**

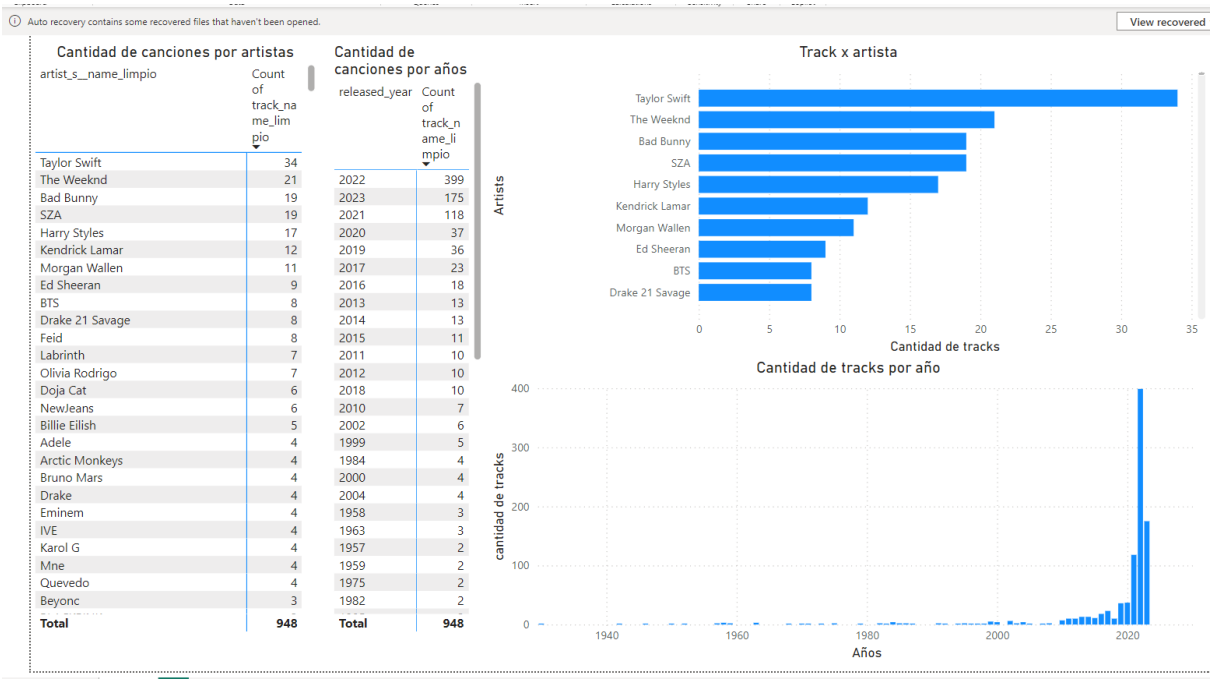
Utilizando el comando WITH, creamos una tabla temporal para calcular el total de canciones por artista solista

```
WITH total_songs_artist AS (
  SELECT
    REGEXP_REPLACE(artist_s__name_limpio, r'[^a-zA-Z0-9 ]', '') AS artist_name,
    COUNT(track_id) AS total_songs
  FROM
    `mjproyecto2.database.view_ts_tc_tt`
  WHERE
    artist_count = 1
  GROUP BY
    artist_name
)
SELECT
  *
FROM
  total_songs_artist
ORDER BY
  total_songs DESC;
```

Row	artist_name	total_songs
1	Taylor Swift	34
2	The Weeknd	21
3	Bad Bunny	19
4	SZA	19
5	Harry Styles	17
6	Kendrick Lamar	12
7	Morgan Wallen	11
8	Ed Sheeran	9
9	Feid	8
10	BTS	8
11	Olivia Rodrigo	7
12	Labrinth	7
13	NewJeans	6
14	Doja Cat	6
15	Billie Eilish	5
16	Arctic Monkeys	4
17	Eminem	4
18	Adele	4

Análisis exploratorio:

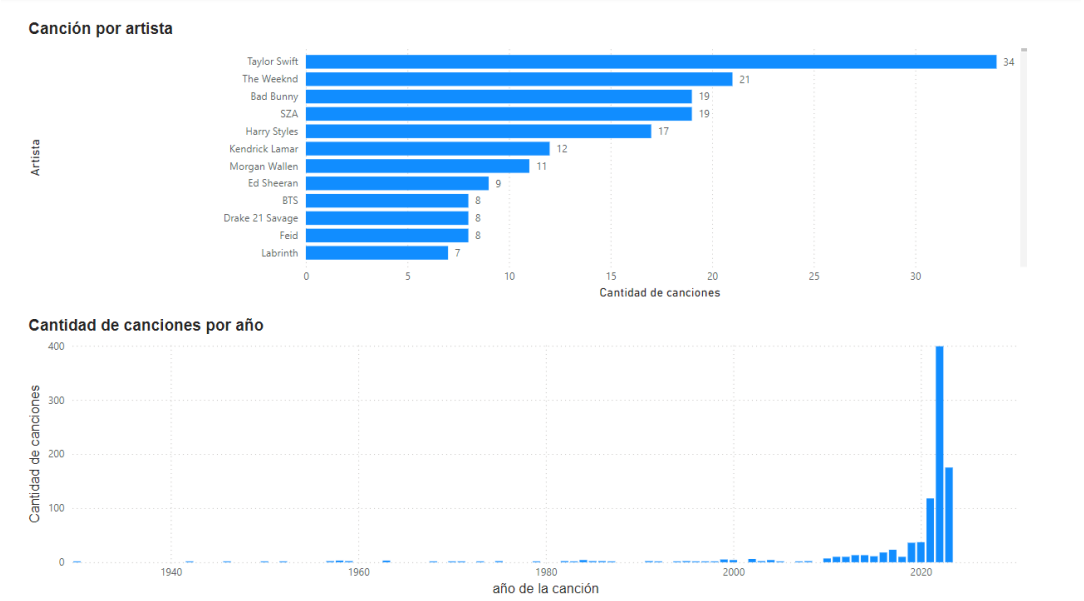
1. **Agrupar datos según variables categóricas.**



Agrupamos los datos importados de BigQuery a Power Bi para crear dos variables categóricas, las cuales fueron expuesta en una tabla “Matrix” y gráficos .

2. **Visualizar las variables categóricas.**

Aquí tenemos la visualización de las gráficas un poco más grande:



3. **Aplicar medidas de tendencia central.**

A través de tablas en Power BI, debemos calcular las medidas de tendencia central (Promedio y Mediana).

- Aquí tenemos el cálculo del bpm: cuando estamos creando la matriz, en la celda de valores colocamos 4 veces la columna de “bpm” y le asignamos a cada uno el valor que debe reflejar.

  Average = promedio; mediana, mínimo y máximo.

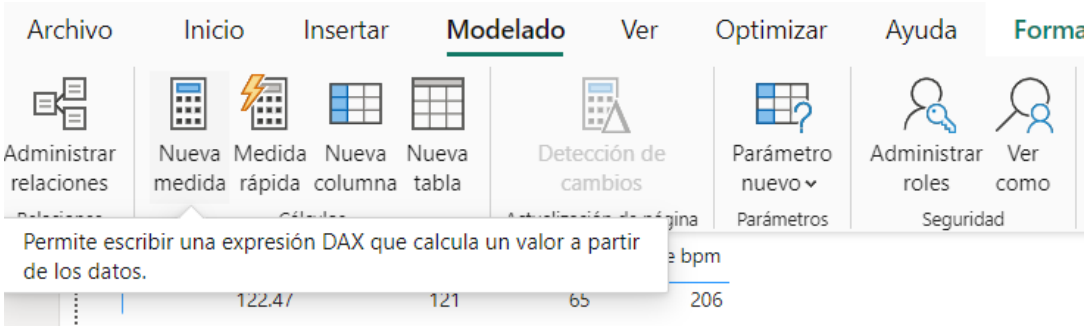
Promedio de bpm	Mediana de bpm	Mín. de bpm	Máx. de bpm
122.47	121	65	206

- Aquí el cálculo de streams:

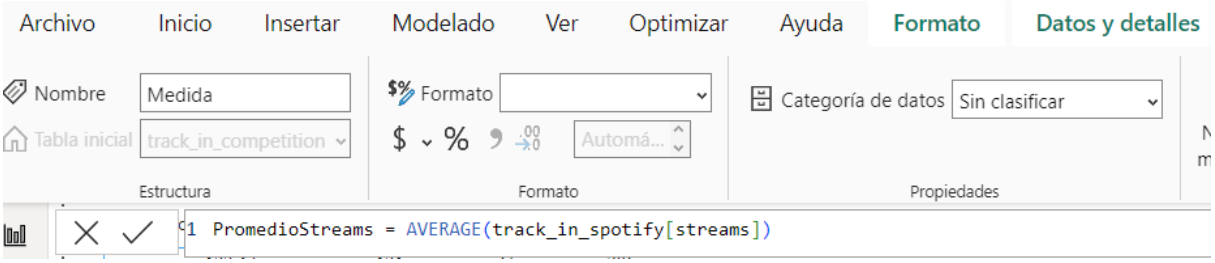
Promedio de streams	514336542.00
Mediana de streams	289165139
Mín. de streams	2762
Máx. de streams	3703895074

para la matriz de streams, se ve visualmente desordenado ya que los números no cuentan con puntos. Así que se realizó la tabla de otra manera:

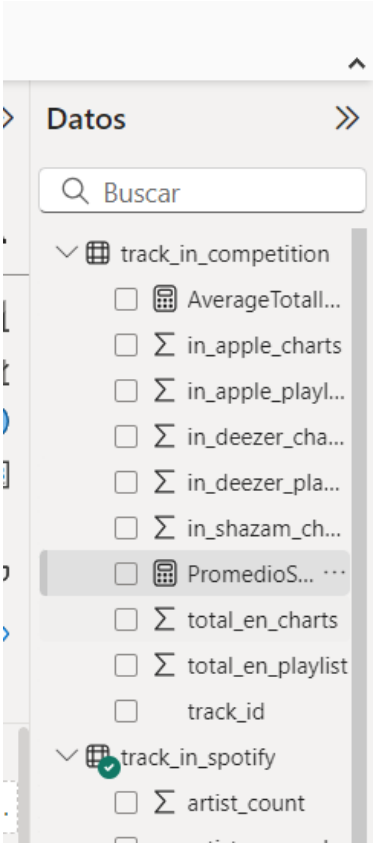
a) Fuimos a Modelado y seleccionamos “nueva medida”



b) Colocamos el nombre de la fórmula como: “PromedioStreams” o “AverageStreams”, colocamos la fórmula y la guardamos.



c) La fórmula aparecerá en la columnas Datos :



d) Creamos una nueva matriz y arrastramos esa fórmula a la matriz y aparecerá reflejado el dato obtenido con los puntos:

AverageStreams
514,336,542.00

Este proceso se debe volver a repetir para calcular el promedio y la media de cada dato y se vea reflejado con puntos, ya que si se hace directamente editando la información en la columna "visualizaciones" será más rápido pero no se obtienen los puntos.

PromedioTotalEnPlaylist	MedianaTotalEnPlaylist
455.14	86.00

esta sería el promedio y la mediana de "total_en_playlist". Las fórmulas utilizadas fueron:

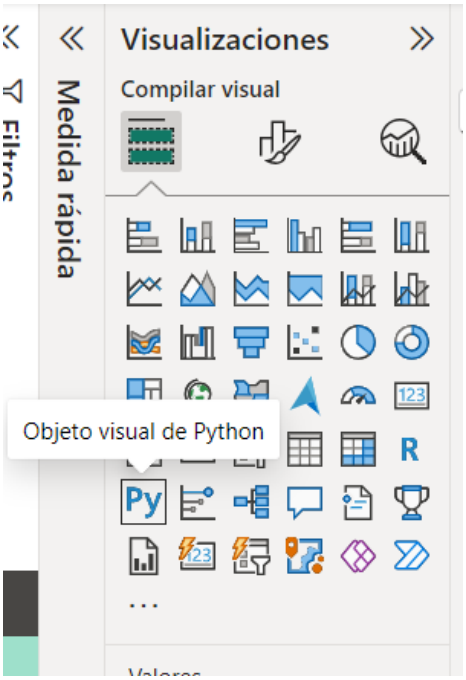
PromedioTotalEnPlaylist = AVERAGE(track_in_competition[total en playlist])

MedianaTotalEnPlaylist = MEDIAN(track_in_competition[total en playlist])

4. **Visualizar distribución.**

Aquí trabajaremos con Python para visualizar las variables con un histograma.

a) La variable de bpm: primero seleccionaremos "Py" en la sección de visualizaciones.

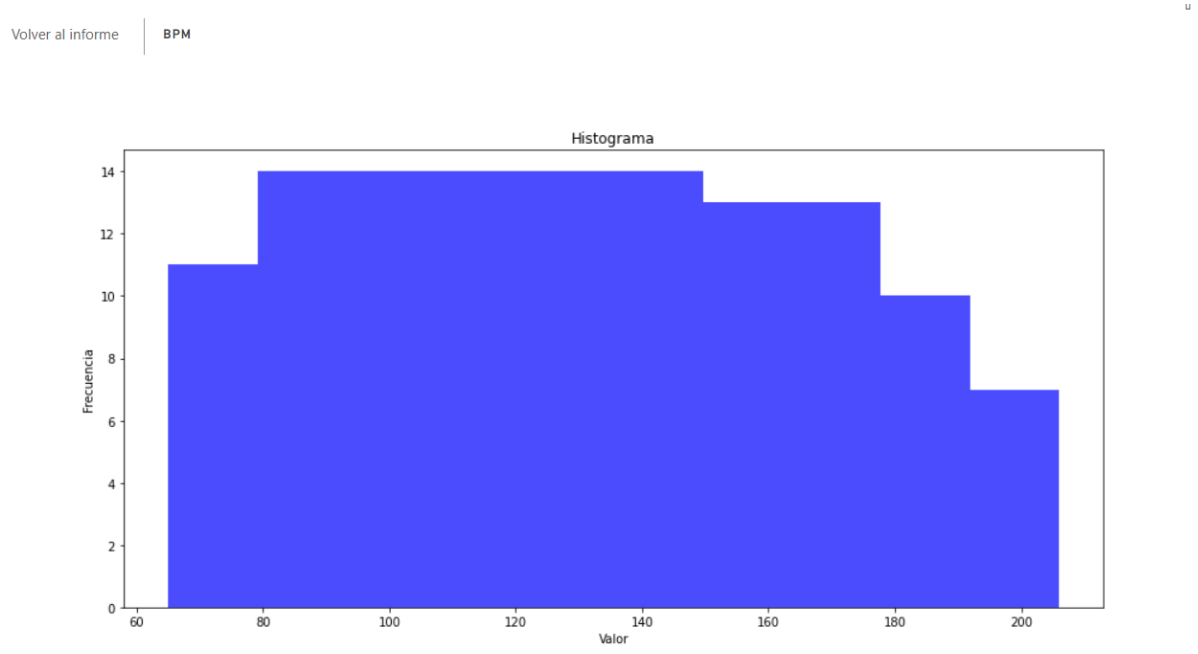


En el espacio que nos aparecerá debajo de la gráfica:

colocaremos el código:

```
import matplotlib.pyplot as plt
import pandas as pd

# Los datos seleccionados en Power BI se pasan automáticamente al dataframe 'data
data = dataset['bpm']  # Asegúrate de que 'bpm' es el nombre correcto de la colum

# Crea el histograma
plt.hist(data, bins=10, color='blue', alpha=0.7)

# Agrega etiquetas y título
plt.xlabel('Valor')
plt.ylabel('Frecuencia')
plt.title('Histograma')

# Muestra el histograma
plt.show()
```

Corremos el programa y aparecerá esto:

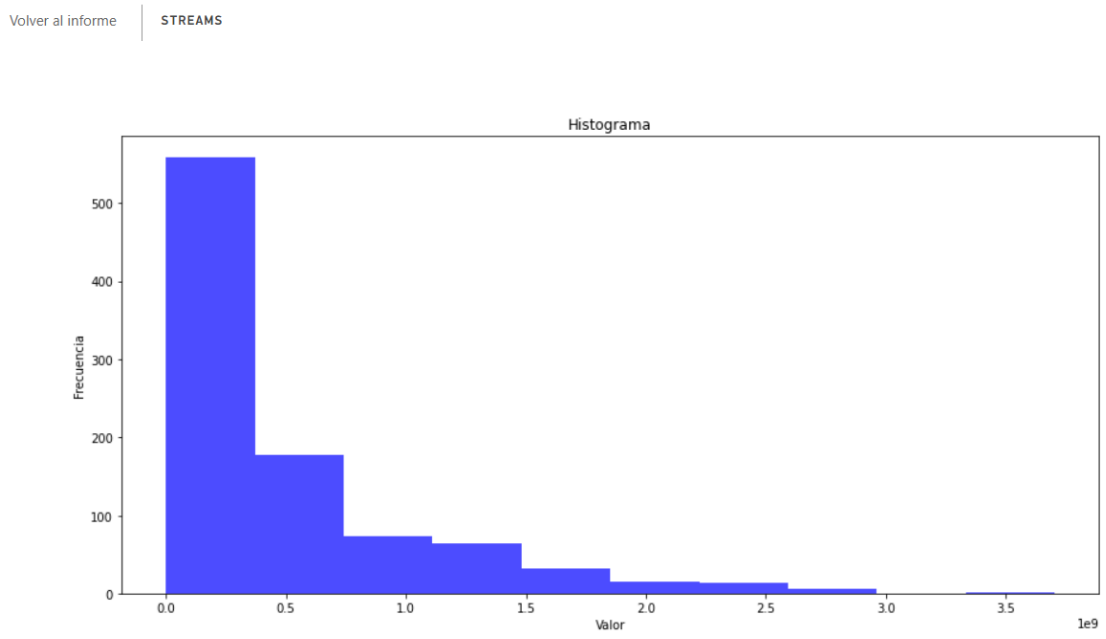


Este mismo paso lo realizaremos con las demás variables.

b) Código de la variable de streams:

```
import matplotlib.pyplot as plt
import pandas as pd

# Los datos seleccionados en Power BI se pasan automáticamente al dataframe 'data
data = dataset['streams']  # Asegúrate de que 'bpm' es el nombre correcto de la c

# Crea el histograma
plt.hist(data, bins=10, color='blue', alpha=0.7)

# Agrega etiquetas y título
plt.xlabel('Valor')
plt.ylabel('Frecuencia')
plt.title('Histograma')

# Muestra el histograma
plt.show()
```

Histograma de streams:



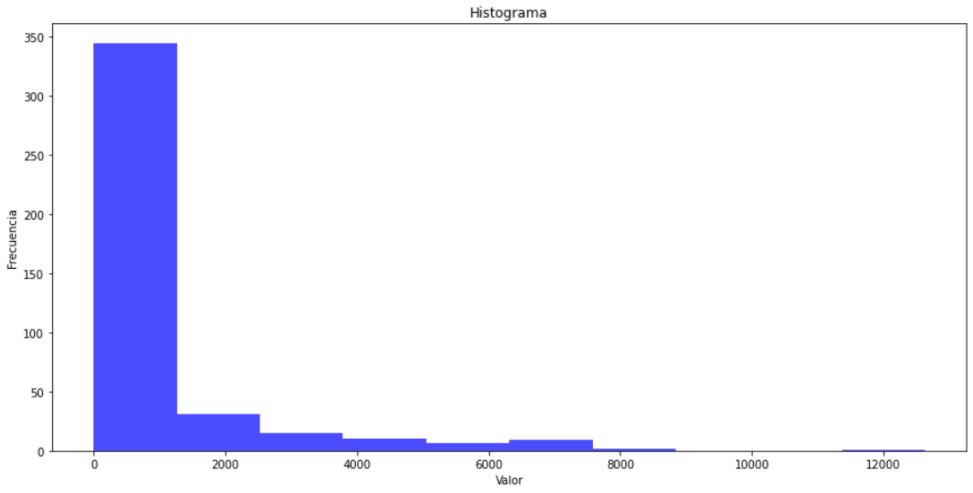
b) Código de la variable de total_en_playlist:

```
import matplotlib.pyplot as plt
import pandas as pd

# Los datos seleccionados en Power BI se pasan automáticamente al dataframe 'data
data = dataset['total_en_playlist']  # Asegúrate de que 'bpm' es el nombre correc

# Crea el histograma
plt.hist(data, bins=10, color='blue', alpha=0.7)
```

```
# Agrega etiquetas y título
plt.xlabel('Valor')
plt.ylabel('Frecuencia')
plt.title('Histograma')

# Muestra el histograma
plt.show()
```

Histograma de total_en_playlist:

/olver al informe | TOTAL_EN_PLAYLIST



5. **Aplicar medidas de dispersión.**

Calcularemos las medidas de dispersión a través de la desviación estándar.

a) bpm:

Promedio de bpm	Mediana de bpm	Mín. de bpm	Máx. de bpm	Desviación estándar de bpm
122.47	121	65	206	28.03

💡 Conclusión

Los valores de bpm analizados muestran una diversidad rítmica notable en la competición. La proximidad entre la mediana y el promedio indica una distribución equilibrada, donde no predominan excesivamente ni los ritmos rápidos ni los lentos. La variabilidad de los ritmos, con una desviación

estándar de 28.03 bpm, sugiere que la selección musical abarca una amplia gama de géneros, desde baladas lentas hasta ritmos frenéticos.

Esta diversidad es clave para mantener el interés y la emoción del público, proporcionando una experiencia musical rica y dinámica. En futuras competiciones, mantener esta variabilidad rítmica podría ser beneficioso para atraer y satisfacer a un público amplio, asegurando momentos de calma y reflexión así como explosiones de energía y entusiasmo. La selección de pistas debe continuar reflejando esta variedad para maximizar el impacto y la satisfacción del evento.

b) streams:

AverageStreams	MedianaStreams	MínimoStreamsFormateado	MaximoStreamsFormateado	DesviacionEstandarStreamsFormateada
514,336,542.00	289,165,138.50	2.762	3.703.895.074	567.497.594,57

Conclusión

Los datos de streams proporcionan una imagen clara de la distribución y variabilidad en la popularidad de las pistas. La gran diferencia entre la mediana y el promedio, junto con una desviación estándar alta, indica una distribución sesgada donde pocas pistas extremadamente populares elevan el promedio. Esto sugiere que la mayoría de las pistas tienen un rendimiento moderado en cuanto a streams, mientras que unas pocas alcanzan niveles de popularidad muy altos.

Esta variabilidad puede deberse a múltiples factores como el género musical, la popularidad del artista, estrategias de marketing y la presencia en listas de reproducción populares. Entender esta distribución es crucial para identificar tendencias y planificar estrategias para aumentar la visibilidad y el rendimiento de las pistas menos populares. Además, los valores extremos (tanto mínimos como máximos) muestran que hay una significativa disparidad en el éxito de las pistas, lo cual es común en la industria de la música.

Para futuras estrategias, podría ser útil enfocarse en analizar las características de las pistas más exitosas y aplicar esas estrategias a otras pistas para mejorar su rendimiento. Esto incluye optimizar las campañas de marketing, colaboraciones estratégicas y mayor presencia en plataformas y listas de reproducción clave.

c) total_en_playlist:

PromedioTotalEnPlaylist	MedianaTotalEnPlaylist	Mín. de total_en_playlist	MaximoTotalenplaylistFormateado	DesviacionEstandarTotalenplaylistFormateada
455.14	86.00	0	12.632	1.175,92

Conclusión

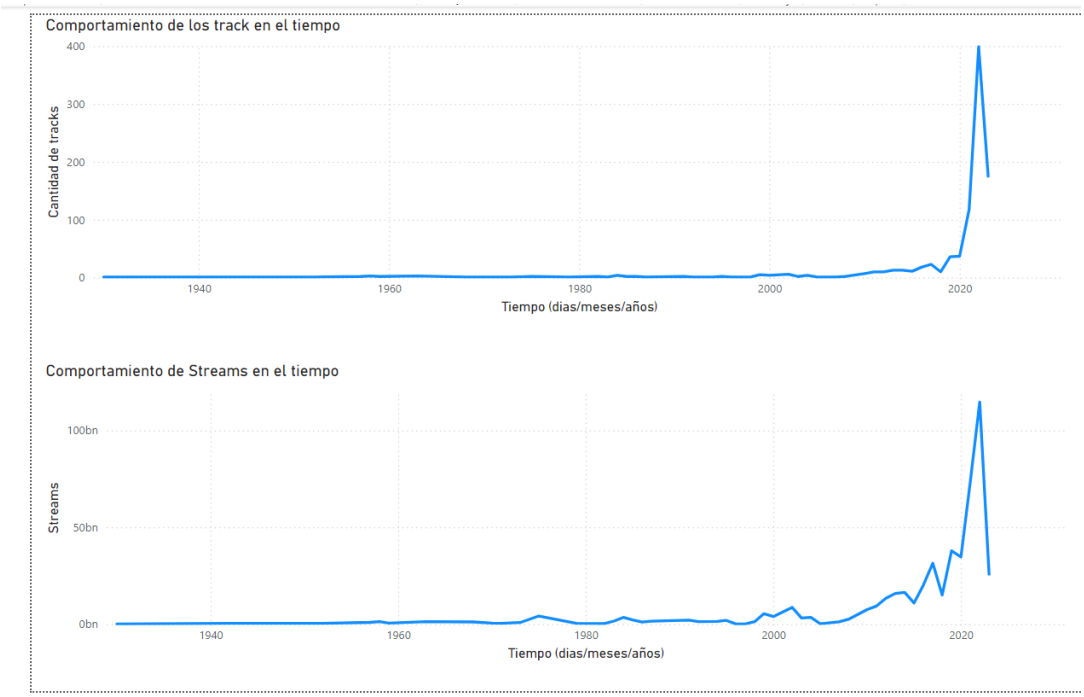
Los datos proporcionados revelan una gran diversidad en la inclusión de pistas en playlists. La diferencia significativa entre la mediana (86.00) y el promedio (455.14) junto con una alta desviación estándar (1,175.92) indica que la distribución está sesgada. Esto significa que mientras algunas pistas están incluidas en un número muy alto de playlists, la mayoría de las pistas tienen una inclusión mucho menor.

La inclusión en playlists es crucial para el éxito en plataformas de streaming, ya que aumenta la visibilidad y la probabilidad de obtener más streams. Los valores mínimos de 0 muestran que hay pistas que no han logrado inclusión en ninguna playlist, lo cual puede ser un área de oportunidad para mejorar la promoción y la estrategia de marketing.

En resumen, aunque algunas pistas ya disfrutan de una alta inclusión en playlists, hay un margen considerable para mejorar la visibilidad de otras pistas mediante estrategias de marketing más focalizadas y colaboraciones estratégicas con curadores de playlists.

6. **Visualizar el comportamiento de los datos a lo largo del tiempo.**

Visualizar a través de gráficos de líneas, el comportamiento de los datos a lo largo del tiempo en Power BI:



Comportamiento de los tracks en el tiempo:

- **Aumento reciente significativo**: Se observa un aumento drástico en la cantidad de tracks alrededor del año 2020. Esto podría indicar un auge en la producción o disponibilidad de tracks en ese período.
- **Estabilidad previa**: Antes del aumento reciente, la cantidad de tracks se mantuvo bastante estable durante varias décadas, con solo pequeños incrementos ocasionales.
- **Rango temporal amplio**: El gráfico abarca un periodo largo de tiempo, desde antes de 1940 hasta 2020, proporcionando una perspectiva histórica amplia sobre la cantidad de tracks.
- **Pico y descenso**: Después del pico significativo alrededor de 2020, hay una caída notable en la cantidad de tracks. Esto podría sugerir una corrección después de un pico anormalmente alto.

Comportamiento de los streams en el tiempo

Crecimiento exponencial reciente: Similar al gráfico anterior, hay un aumento dramático en la cantidad de streams alrededor del año 2020, sugiriendo un auge en la popularidad o el uso de plataformas de streaming en ese período.

Estabilidad anterior: Antes del aumento exponencial, la cantidad de streams se mantuvo relativamente estable durante muchas décadas, con pequeños incrementos ocasionales.

Pico y descenso: Al igual que con la cantidad de tracks, hay un pico pronunciado en la cantidad de streams seguido de una caída significativa después del punto máximo.

Cambio en la tendencia: A partir del año 2000, se observa un incremento gradual en la cantidad de streams, que se acelera significativamente a partir de 2010, lo que podría coincidir con la popularización de servicios de streaming de música y otros contenidos.

7. **Calcular cuartiles, deciles o percentiles.**
   a. **Aplicación de cuartiles**

JOB INFORMATION | RESULTS | CHART

Row	streams	quartiles_streams
1	94186466	1
2	115010040	1
3	175399345	2
4	190625045	2
5	245095641	2
6	304079786	3
7	349746291	3
8	421040617	3
9	497225336	3
10	1284942608	4
11	305650299	3
12	477033549	3
13	510876816	3
14	2665343922	4

```
SELECT
    streams,
    NTILE(4) OVER (ORDER BY streams) AS quartiles_streams
FROM
    `mjproyecto2.database.view_ts_tc_tt`
```

NTILE: me permite calcular un número de datos

OVER: sobre que variable voy a dividir

ORDEN BY: ordenas los datos para poder aplicar los cuartiles

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	track_id	track_name_limpio	artist_s__name_limpio	streams	quartiles_streams
1	1406287	I Wanna Be Yours	Arctic Monkeys	1297026226	4
2	3921634	Blank Space	Taylor Swift	1355959075	4
3	4796316	Style	Taylor Swift	786181836	4
4	4527090	Yellow	Chris Molitor	1755214421	4
5	3126367	Viva La Vida	Coldplay	1592909789	4
6	8502696	Riptide	Vance Joy	2009094673	4
7	1216337	Summertime Sadness	Lana Del Rey	983637508	4
8	1729982	Lover	Taylor Swift	882831184	4
9	2394886	Mockingbird	Eminem	1241559043	4
10	4350729	Set Fire to the Rain	Adele	1163620694	4

```
WITH Quartiles AS (
    SELECT
        streams,
```

```
        NTILE(4) OVER (ORDER BY streams) AS quartiles_streams
    FROM
        `mjproyecto2.database.view_ts_tc_tt`
)
SELECT
  a. *,
  Quartiles.quartiles_streams
  FROM `mjproyecto2.database.view_ts_tc_tt` a

  LEFT JOIN Quartiles

  ON a.streams=Quartiles.streams
```

WITH: para crear esta nueva variables ocuparemos en nuestra VIEW de uso

LEFT JOIN: unimos esta nueva columna a la tabla view.

Este tipo de consulta nos ayudará a analizar cada característica que deseamos comparar, asignándole un "puntaje" para su segmentación.

b. Aplicación de IF

```
WITH
  Quartiles AS (
  SELECT
    streams,
    NTILE(4) OVER (ORDER BY streams) AS quartiles_streams
  FROM
    `mjproyecto2.database.view_ts_tc_tt` )
SELECT
  a. *,
 IF(Quartiles.quartiles_streams = 4, "alto", "bajo") AS categ_streams
FROM
  `mjproyecto2.database.view_ts_tc_tt` a
LEFT JOIN
  Quartiles
ON
  a.streams=Quartiles.streams
```

Row	track_id	track_name_limpio	artist_s__name_limpio	streams	categ_streams
1	1406287	I Wanna Be Yours	Arctic Monkeys	1297026226	alto
2	3921634	Blank Space	Taylor Swift	1355959075	alto
3	4796316	Style	Taylor Swift	786181836	alto
4	4527090	Yellow	Chris Molitor	1755214421	alto
5	3126367	Viva La Vida	Coldplay	1592909789	alto
6	8502696	Riptide	Vance Joy	2009094673	alto
7	1216337	Summertime Sadness	Lana Del Rey	983637508	alto
8	1729982	Lover	Taylor Swift	882831184	alto
9	2394886	Mockingbird	Eminem	1241559043	alto
10	4350729	Set Fire to the Rain	Adele	1163620694	alto
11	7811952	Money Trees	Kendrick Lamar Jay Rock	1093605526	alto
12	4458161	Radio	Lana Del Rey	284819874	bajo
13	3253565	Perfect	Ed Sheeran	2559529074	alto

Si quisiéramos agregar más de una calificación podemos hacerlo de dos maneras, con IF o CASE

- **IF**

```
WITH
  Quartiles AS (
    SELECT
      streams,
      NTILE(4) OVER (ORDER BY streams) AS quartiles_streams
    FROM
      `mjproyecto2.database.view_ts_tc_tt`
  )
SELECT
  a.*,
  IF(Quartiles.quartiles_streams = 1, 'bajo',
    IF(Quartiles.quartiles_streams = 2, 'medio bajo',
      IF(Quartiles.quartiles_streams = 3, 'medio alto', 'alto'))) AS categ_stream
FROM
  `mjproyecto2.database.view_ts_tc_tt` a
LEFT JOIN
  Quartiles
ON
  a.streams = Quartiles.streams;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	track_id	track_name_limpio	artist_s__name_limpio	streams	categ_streams
12	4458161	Radio	Lana Del Rey	284819874	medio bajo
13	3253565	Perfect	Ed Sheeran	2559529074	alto
14	1663974	Without Me	Eminem	1687664027	alto
15	3412305	Somewhere Only We Know	Keane	1089402494	alto
16	3766692	Danza Kuduro	Don Omar Lucenzo	1279434863	alto
17	7325433	Counting Stars	OneRepublic	2011464183	alto
18	6735267	Enchanted	Taylor Swift	621660989	medio alto
19	4242212	We Found Love	Rihanna Calvin Harris	1235005533	alto
20	2664577	Do I Wanna Know	Arctic Monkeys	1788326445	alto
21	8580552	Demons	Imagine Dragons	1840364617	alto
22	2031309	Shake It Off	Taylor Swift	1113838873	alto
23	5999455	Whyd You Only Call Me When Y...	Arctic Monkeys	1267333350	alto
24	2008299	Just The Way You Are	Bruno Mars	1641426668	alto
25	2670198	The Real Slim Shady	Eminem	1424589568	alto
26	6161550	Cupid Twin Ver FIFTY FIFTY S...	sped up 8282	103762518	bajo
27	6766942	Lose Yourself Soundtrack Vers...	Eminem	1829992958	alto

- **CASE**

```
WITH
  Quartiles AS (
    SELECT
      streams,
      NTILE(4) OVER (ORDER BY streams) AS quartiles_streams
    FROM
      `mjproyecto2.database.view_ts_tc_tt`
  )
SELECT
  a.*,
  CASE
    WHEN Quartiles.quartiles_streams = 1 THEN 'bajo'
    WHEN Quartiles.quartiles_streams = 2 THEN 'medio bajo'
```

```
    WHEN Quartiles.quartiles_streams = 3 THEN 'medio alto'
    WHEN Quartiles.quartiles_streams = 4 THEN 'alto'
  END AS categ_streams
FROM
  `mjproyecto2.database.view_ts_tc_tt` a
LEFT JOIN
  Quartiles
ON
  a.streams = Quartiles.streams;
```

Row	track_id	track_name_limpio	artist_s__name_limpio	...	streams	...	categ_streams
12	4458161	Radio	Lana Del Rey		284819874		medio bajo
13	3253565	Perfect	Ed Sheeran		2559529074		alto
14	1663974	Without Me	Eminem		1687664027		alto
15	3412305	Somewhere Only We Know	Keane		1089402494		alto
16	3766692	Danza Kuduro	Don Omar Lucenzo		1279434863		alto
17	7325433	Counting Stars	OneRepublic		2011464183		alto
18	6735267	Enchanted	Taylor Swift		621660989		medio alto
19	4242212	We Found Love	Rihanna Calvin Harris		1235005533		alto
20	2664577	Do I Wanna Know	Arctic Monkeys		1788326445		alto
21	8580552	Demons	Imagine Dragons		1840364617		alto
22	2031309	Shake It Off	Taylor Swift		1113838873		alto
23	5999455	Whyd You Only Call Me When Y...	Arctic Monkeys		1267333350		alto
24	2008299	Just The Way You Are	Bruno Mars		1641426668		alto
25	2670198	The Real Slim Shady	Eminem		1424589568		alto
26	6161550	Cupid Twin Ver FIFTY FIFTY S...	sped up 8282		103762518		bajo
27	6766942	Lose Yourself Soundtrack Vers...	Eminem		1829992958		alto

Ambas versiones logran el mismo objetivo, pero la versión con `CASE` es generalmente más clara y fácil de leer cuando se tienen múltiples condiciones. Pero para este caso, ocuparemos IF para continuar con las instrucciones.

Calculando los cuartiles de las demás categorías, se nota una inconsistencia en donde el total de usuarios se altera:

1 – 50 de 954

modificamos el código para no perder el monto total de los usuarios que es 948.

```
WITH quartiles AS (
    SELECT
    track_id,
        streams,
        NTILE(4) OVER (ORDER BY streams) AS quartile_streams
    FROM
        `proyecto-2-hipotesis-426800.spotify_2023.vista_track_data`
)

, categorias AS (
    SELECT
        track_id,
        streams,
        IF(quartile_streams = 1, 'Poco bailable',
```

```
            IF(quartile_streams = 2, 'Moderadamente bailable',
                IF(quartile_streams = 3, 'Bastante bailable', 'Muy bailable')
            )
        ) AS categoria_streams
    FROM
        quartiles
)

SELECT
    t.*,
    c.categoria_streams
FROM
    `proyecto-2-hipotesis-426800.spotify_2023.vista_track_data` t
LEFT JOIN
    categorias c
ON
    t.track_id = c.track_id
ORDER BY
    t.streams;
```

Con esta fórmula iremos trabajando con las siguiente información que tenemos calculando el quartil y crear categorías:

streams
total_en_playlist
bpm
danceability_%
valence_%
energy_%
acousticness_%
instrumentalness_%
liveness_%
speechiness_%
fecha de lanzamiento

Se tomó la decisión de trabajar con cuartiles y manejar la clasificación en 2 (y no 4) para hacer más sencillo el análisis.

```
WITH categorias AS (
    SELECT
        track_id,
        streams,
        total_en_playlists,
        bpm,
        `danceability_%`,
```

```
        `valence_%`,
        `energy_%`,
        `acousticness_%`,
        `instrumentalness_%`,
        `liveness_%`,
        `speechiness_%`,
        fecha_completa,

        NTILE(2) OVER (ORDER BY streams) AS quartiles_streams,
        NTILE(2) OVER (ORDER BY total_en_playlists) AS quartiles_tp,
        NTILE(2) OVER (ORDER BY bpm) AS quartiles_bpm,

        CASE
            WHEN `danceability_%` <= 50 THEN 1
            ELSE 2
        END AS quartiles_dance,

        CASE
            WHEN `valence_%` <= 50 THEN 1
            ELSE 2
        END AS quartiles_valence,

        CASE
            WHEN `energy_%` <= 50 THEN 1
            ELSE 2
        END AS quartiles_energy,

        CASE
            WHEN `acousticness_%` <= 50 THEN 1
            ELSE 2
        END AS quartiles_acousticness,

        CASE
            WHEN `instrumentalness_%` <= 50 THEN 1
            ELSE 2
        END AS quartiles_instrumentalness,

        CASE
            WHEN `liveness_%` <= 50 THEN 1
            ELSE 2
        END AS quartiles_liveness,

        CASE
            WHEN `speechiness_%` <= 50 THEN 1
            ELSE 2
        END AS quartiles_speechiness,

        NTILE(2) OVER (ORDER BY fecha_completa) AS quartiles_fc
    FROM
        `proyecto-2-hipotesis-426800.spotify_2023.vista_track_data`
)
```

```
SELECT
    t.*,
    CASE c.quartiles_streams
        WHEN 1 THEN 'bajo'
        ELSE 'alto'
    END AS categoria_streams,

    CASE c.quartiles_tp
        WHEN 1 THEN 'bajo'
        ELSE 'alto'
    END AS categoria_total_en_playlists,

    CASE c.quartiles_bpm
        WHEN 1 THEN 'bajo'
        ELSE 'alto'
    END AS categoria_bpm,

    CASE c.quartiles_dance
        WHEN 1 THEN 'poco bailable'
        ELSE 'bailable'
    END AS danceability_index,

    CASE c.quartiles_valence
        WHEN 1 THEN 'negativa/melancólica'
        ELSE 'positiva/alegre'
    END AS valence_index,

    CASE c.quartiles_energy
        WHEN 1 THEN 'poco enérgico'
        ELSE 'enérgico'
    END AS energy_index,

    CASE c.quartiles_acousticness
        WHEN 1 THEN 'bajo nivel acústico'
        ELSE 'alto nivel acústico'
    END AS acousticness_index,

    CASE c.quartiles_instrumentalness
        WHEN 1 THEN 'bajo nivel instrumental'
        ELSE 'alto nivel instrumental'
    END AS instrumentalness_index,

    CASE c.quartiles_liveness
        WHEN 1 THEN 'poca presencia de público'
        ELSE 'alta presencia de público'
    END AS liveness_index,

    CASE c.quartiles_speechiness
        WHEN 1 THEN 'poca presencia de habla'
        ELSE 'alta presencia de habla'
```

```
    END AS speechiness_index,

    CASE c.quartiles_fc
        WHEN 1 THEN 'canción vieja'
        ELSE 'canción actual'
    END AS tiempo_en_el_mercado

FROM
    `proyecto-2-hipotesis-426800.spotify_2023.vista_track_data` t
LEFT JOIN
    categorias c ON t.track_id = c.track_id
ORDER BY
    t.track_id
```

Es extensa la fórmula pero de esta manera debemos aplicar para cada categoría que presenta valores atípicos, para que los datos no presenten alteraciones y poder manejar mejor el análisis.

💡 EJEMPLO DE DATO ATÍPICO

Este es un ejemplo de los valores atípicos que se presentaba en `instrumentalness_%` ya que existe un sesgo hacia valores bajos (que en su mayoría son 0) y eso altera la distribución al calcular los cuartiles.

cuartiles_ordenados

Vista previa descargada el Friday, June 28, 2024

ousticness_index	instrumentalness_%	instrumentalness_index	liven
ıjo nivel acústico	0	Bajo nivel instrumental	
uy bajo nivel acústico	0	Alto nivel instrumental	
uy alto nivel acústico	0	Alto nivel instrumental	
uy bajo nivel acústico	44	Muy alto nivel instrumental	
ıjo nivel acústico	0	Muy alto nivel instrumental	
to nivel acústico	0	Alto nivel instrumental	
to nivel acústico	6	Muy alto nivel instrumental	
uy alto nivel acústico	72	Muy alto nivel instrumental	
ıjo nivel acústico	0	Alto nivel instrumental	
ıjo nivel acústico	0	Bajo nivel instrumental	
to nivel acústico	0	Muy alto nivel instrumental	
uy alto nivel acústico	0	Alto nivel instrumental	
uy bajo nivel acústico	0	Muy bajo nivel instrumental	
uy alto nivel acústico	0	Muy bajo nivel instrumental	

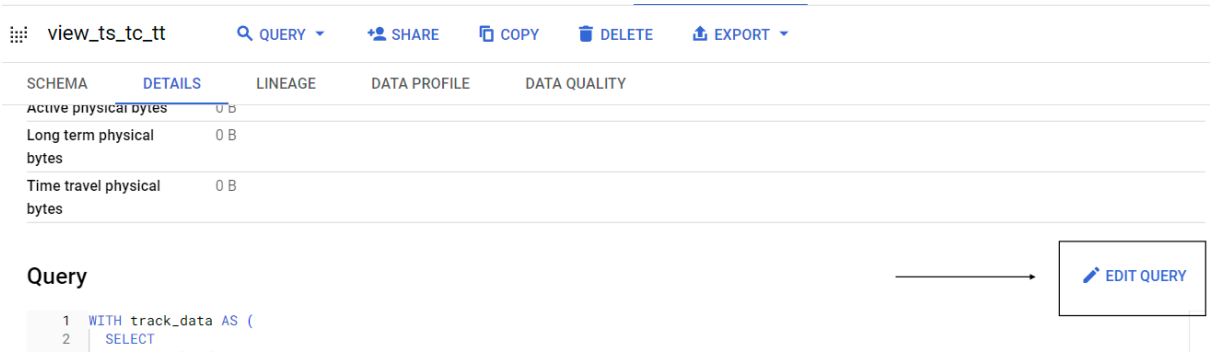
por esta razón fue que se hizo la especificación en la fórmula.

Recomendado ordenar las columnas para poder tener facilidad al visualizar los resultados, lo haremos de forma manual:

```
CREATE VIEW `proyecto-2-hipotesis-426800.spotify_2023.cuartiles_cuartiles_ordenad
SELECT
track_id,
track_name_limpio,
artists_name_limpio,
artist_count,
released_year,
released_month,
released_day,
fecha_completa,
tiempo_en_el_mercado,
in_spotify_playlists,
in_apple_playlists,
in_deezer_playlists,
total_en_playlists,
in_spotify_charts,
in_apple_charts,
in_deezer_charts,
in_shazam_charts,
total_en_charts,
streams,
categoria_streams,
bpm,
categoria_bpm,
`danceability_%`,
danceability_index,
`valence_%`,
valence_index,
`energy_%`,
energy_index,
`acousticness_%`,
acousticness_index,
`instrumentalness_%`,
instrumentalness_index,
`liveness_%`,
liveness_index,
`speechiness_%`,
speechiness_index

from `proyecto-2-hipotesis-426800.spotify_2023.cuartiles`
```

Integraremos la información en Power BI para trabajar con las variables recién añadidas. Esto nos permite operar directamente sobre la misma vista sin necesidad de crear una nueva o rehacer el trabajo existente. Para implementarlo, simplemente editamos la vista actual en Power BI, añadimos esta consulta actualizada, y luego procedemos a refrescar los datos para incorporar los cambios.

De preferencia utilizar la tabla ordenada que creamos ya que nos ayudará visualmente al ver los datos.



8. **Calcular correlación entre variables.**

Para aplicar el siguiente paso, fue necesario revisar el listado de track_id, ya que al intentar realizar la correlación, saltaba un error. Gracias a Power Bi, fue más rápido encontrar el "track_id" del problema.

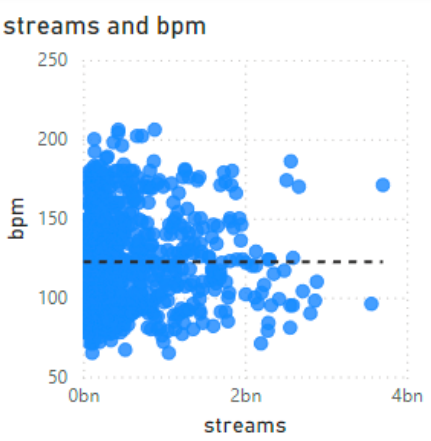
track_id	track_name_limpio	artist_s__name_limpio
0:00	1035	Tisto Tate M

Por ende, antes de comenzar con el paso, es de necesidad limpiarlo, para ello fue necesario agregar el track para ser descartado a nuestro listado de "excluidos" agregamos el dato "0:00" .

```
    `mjproyecto2.database.track_in_spotify` TS
  LEFT JOIN `mjproyecto2.database.track_in_competition` TC
    ON TS.track_id = TC.track_id
  LEFT JOIN `mjproyecto2.database.track_technical_info` TT
    ON TS.track_id = TT.track_id
  WHERE
    SAFE_CAST(TS.streams AS INT64) IS NOT NULL
    AND TS.track_id NOT IN ('5080031', '8173823', '1119309', '3814670','0:00')
    AND TC.track_id NOT IN ('5080031', '8173823', '1119309', '3814670','0:00', '4061483')
    AND TT.track_id NOT IN ('5080031', '8173823', '1119309', '3814670','0:00', '4061483')
),
categorias AS (
  SELECT
```

- Correlación streams vs bpm
  Hipótesis 1:
  "Las canciones con un mayor BPM (Beats Por Minuto) tienen más éxito en términos de streams en Spotify"

Row	correl_streams_bpm
1	-0.00233620358...




```
SELECT
  CORR(streams, bpm) AS correl_streams
FROM
  `mjproyecto2.database.tabla_full_2`
```

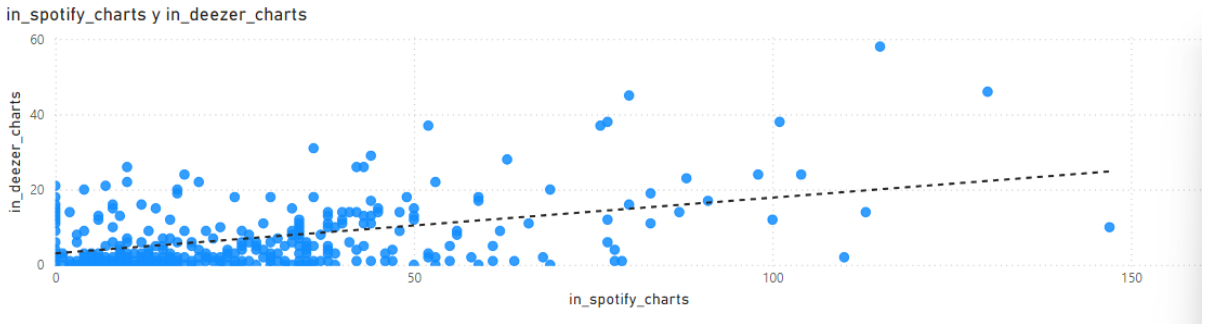
Correlación negativa extremadamente débil. Prácticamente no hay relación lineal entre las variables. Esto significa que prácticamente no hay relación entre las pulsaciones por minuto de una canción y su cantidad de reproducciones.

- Correlación in_spotify_charts vs in_deezer_charts

  Hipótesis 2:
  "Las canciones más populares en el ranking de Spotify también tienen un comportamiento similar en otras plataformas como Deezer"

```
SELECT
CORR(in_spotify_charts, in_deezer_charts) AS cor_spotify_deezer_charts,

FROM
  `proyecto-2-hipotesis-426800.spotify_2023.cuartiles_ordenados`
```

cor_spotify_deezer_c
0.599986055348...



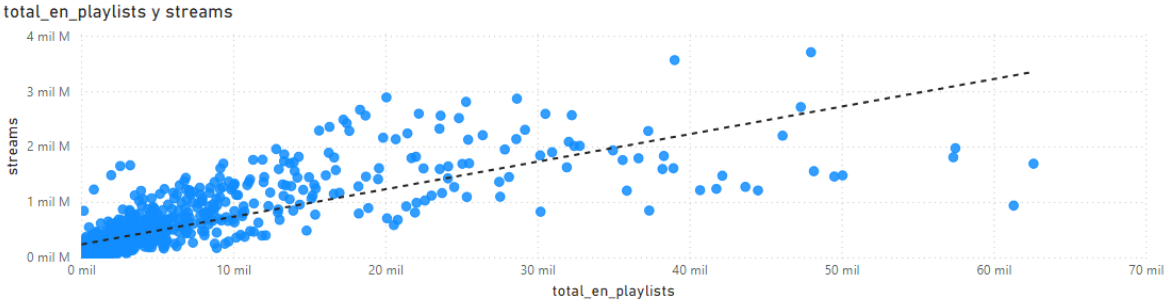
El valor de correlación entre "in_spotify_charts" y "in_deezer_charts" indica una relación lineal positiva moderada entre estas dos variables, lo que sugiere una considerable similitud en las preferencias de los usuarios de ambas plataformas.

- Correlación total_en_playlists vs streams

  Hipótesis 3:
  ”La presencia de una canción en un mayor número de playlists se relaciona con un mayor número de streams”

```
SELECT
CORR(total_en_playlists, streams) AS cor_playlists_streams,

FROM
  `proyecto-2-hipotesis-426800.spotify_2023.cuartiles_ordenados`
```

cor_playlists_stream
0.783568569951...

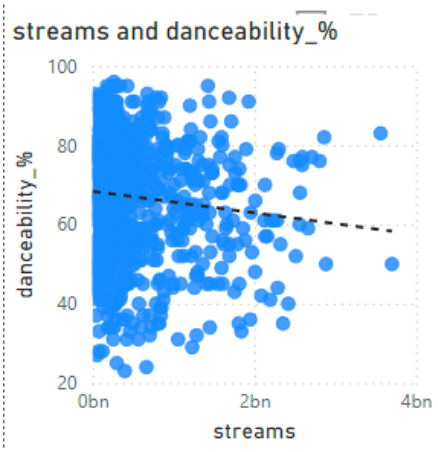


El valor de correlación de 0.783568569 entre "total en playlists" y "streams" indica una relación lineal positiva fuerte entre estas dos variables, lo que sugiere que las canciones que están en más playlists tienden a recibir más streams.

Hipótesis 5:
”Las características de la canción influyen en el éxito en términos de streams en Spotify”

- Correlación streams vs danceability_%

Row	f0_
1	-0.10550116409...

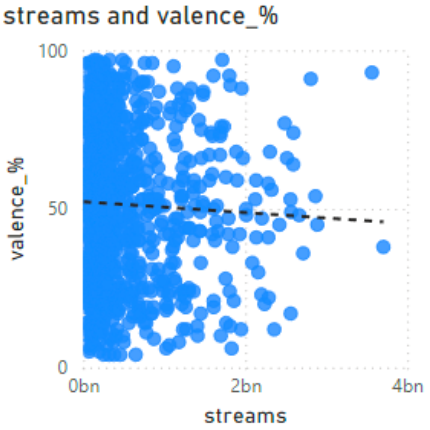



```
SELECT
  CORR(streams, `danceability_%`)
FROM
  `mjproyecto2.database.view_ts_tc_tt
```

Correlación entre “streams” y “danceability” nos muestra que el resultado esta mas cerca del 0, lo que nos indica una relación lineal negativa muy débil entre estas dos variables, lo que sugiere que la capacidad de baile de una canción no es un factor determinante significativo para el número de streams que recibe.

- Correlación streams vs valence_%:

Row	correl_streams_valer
1	-0.04137056374...



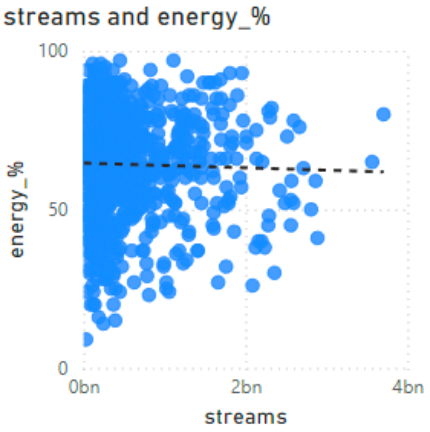

```
SELECT
  CORR(streams, `valence_%`) AS correl
FROM
  `mjproyecto2.database.tabla_full_2`
```

El valor de correlación de -0.04137056374 entre "streams" y "valence_%" indica una relación lineal negativa extremadamente débil entre estas dos variables, lo que sugiere que la positividad o felicidad de una canción no es un factor

determinante significativo para el número de streams que recibe.

- Correlación streams vs energy_%

Row	correl_streams_ener
1	-0.02562682282...

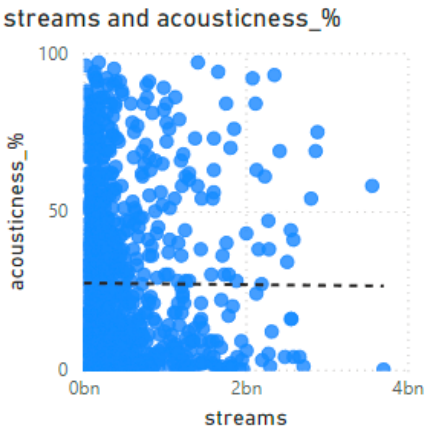



```
SELECT
  CORR(streams,`energy_%` ) AS correl_
FROM
  `mjproyecto2.database.tabla_full_2`
```

El valor de correlación de -0.025626822823 entre "streams" y "energy_%" indica una relación lineal negativa extremadamente débil entre estas dos variables, lo que sugiere que la energía o intensidad de una canción no es un factor determinante significativo para el número de streams que recibe.

- Correlación streams vs acousticness_%

Row	correl_streams_acoustic
1	-0.0052737047472050...

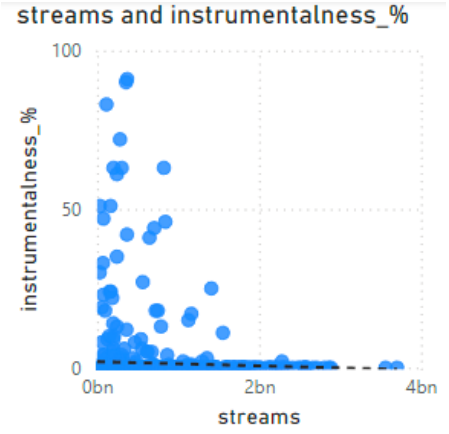



```
SELECT
  CORR(streams,`acousticness_%` ) AS c
FROM
  `mjproyecto2.database.tabla_full_2`
```

El valor de correlación de -0.005273704747 entre "streams" y "acousticness_%" indica una relación lineal negativa extremadamente débil y prácticamente inexistente entre estas dos variables, lo que sugiere que el nivel de elementos acústicos en una canción no es un factor determinante significativo para el número de streams que recibe.

- Correlación streams vs instrumentalness_%

Row	correl_streams_instrun
1	-0.04423448900590...



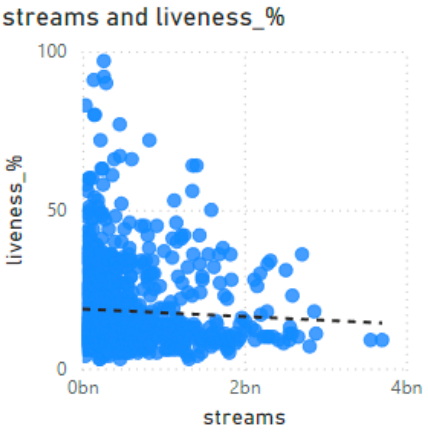

```
SELECT
  CORR(streams,`instrumentalness_%` )
FROM
  `mjproyecto2.database.tabla_full_2`
```

El valor de correlación de -0.044234489005 entre "streams" y "instrumentalness_%" indica una relación lineal negativa extremadamente débil entre estas dos variables, lo que sugiere que la probabilidad de que una canción sea instrumental

no es un factor determinante significativo para el número de streams que recibe.

- Correlación streams vs liveness_%

Row	correl_streams_liveness
1	-0.0494806804902354...

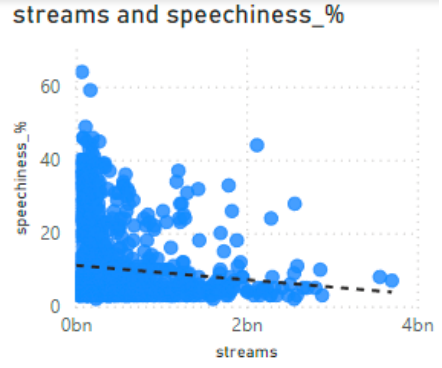



```
SELECT
  CORR(streams,`liveness_%` ) AS corre
FROM
  `mjproyecto2.database.tabla_full_2`
```

El valor de correlación de -0.049480680490 entre "streams" y "liveness_%" indica una relación lineal negativa extremadamente débil entre estas dos variables, lo que sugiere que la presencia de un público en la grabación no es un factor determinante significativo para el número de streams que recibe.

- Correlación streams vs speechiness_%:

Row	correl_streams_speech
1	-0.11276316971291...



```
SELECT
  CORR(streams,`speechiness_%` ) AS co
FROM
  `mjproyecto2.database.tabla_full_2`
```

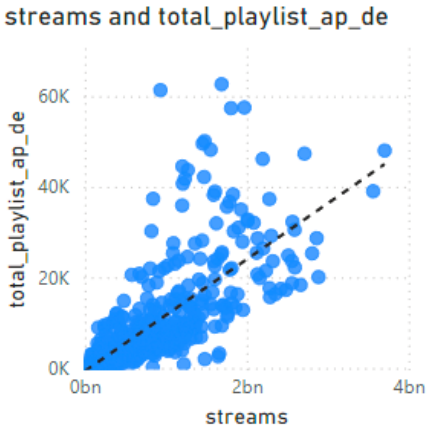
El valor de correlación de -0.11276316971291... entre "streams" y "speechiness_%" indica una relación lineal negativa débil entre estas dos variables, lo que sugiere que la presencia de palabras habladas en una canción tiene una influencia mínima en el número de streams que recibe.

Otras correlaciones:

- Correlación streams vs playlist spotify

```
SELECT CORR(streams, in_spotify_playlists)AS correlacion_stream_tp_ads
FROM
  `mjproyecto2.database.view_ts_tc_tt`
```

Row	correlacion_stream_t
1	0.790364665001...

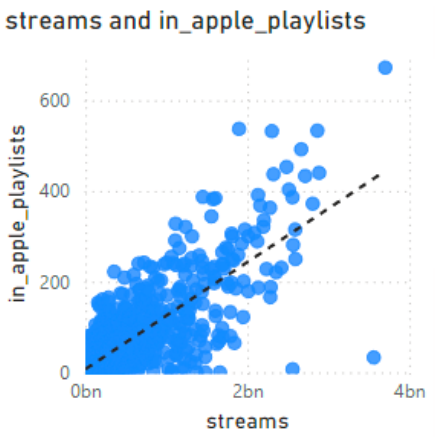


Correlacionamos “streams” (n° total de transmisiones en spotify) con “in_spotify_playlists” (n° de listas de reproduccion de Spotify en las que está la canción). Este resultado esta mas serca del 1, por ende corresponde a una correlación positiva.

- Correlación streams vs playlist apple

```
SELECT
  CORR(streams, in_apple_playlists) AS correl_streams_apple
FROM
  `mjproyecto2.database.tabla_full_2`
```

Row	correl_streams_apple
1	0.77448524315124523

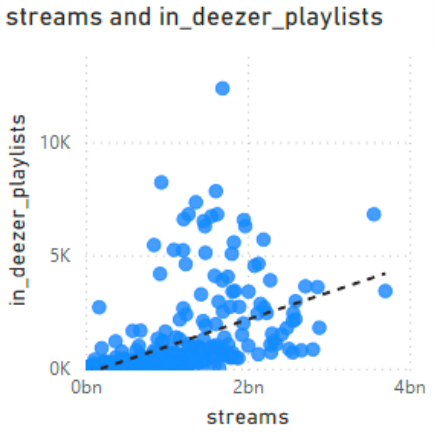


Correlación positiva fuerte. Existe una relación lineal significativa entre las variables.

- Correlación streams vs playlist deezer

```
SELECT
  CORR(streams, in_deezer_playlists) AS correl_streams_deezer
FROM
  `mjproyecto2.database.tabla_full_2`
```

Row	correl_streams_deezer
1	0.59837740705365794



Correlación positiva moderada. Existe una relación lineal entre las variables, pero no es muy fuerte.

- Correlación bpm vs danceability

```
SELECT
  CORR(bpm, `danceability_%` ) AS correl_bpm_dance
FROM
  `mjproyecto2.database.tabla_full_2`
```

Row	correl_bpm_dance
1	-0.14659350776...

Correlación negativa débil. Existe una ligera relación lineal negativa entre las variables

- Correlación energy vs valence

```
SELECT
  CORR(`energy_%`, `valence_%` ) AS correl_energ_valen
FROM
  `mjproyecto2.database.tabla_full_2`
```

Row	correl_energ_valen
1	0.357633210066...

Correlación positiva débil a moderada. Hay una relación lineal positiva, aunque no es muy fuerte

- Correlación acousticness vs instrumentalness:

```
SELECT
  CORR(`acousticness_%`, `instrumentalness_%` ) AS correl_acousti_instrum
FROM
  `mjproyecto2.database.tabla_full_2`
```

Row	correl_acousti_instru
1	0.043841276413...

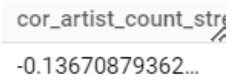
Correlación positiva muy débil. Prácticamente no hay relación lineal entre las variables

- Correlación artist_count vs streams:

```
SELECT
CORR(artist_count, streams) AS cor_artist_count_streams

FROM
  `proyecto-2-hipotesis-426800.spotify_2023.cuartiles_ordenados`
```

El valor de correlación de -0.136708793623 entre "artist_count" y "streams" indica una relación lineal negativa débil entre estas dos variables, lo que sugiere que, aunque hay una ligera tendencia a que más

artistas en una canción se asocien con menos streams, esta tendencia no es fuerte ni significativa.



Aplicar técnica de análisis:

> 💡 **PUNTO INFORMATIVO**

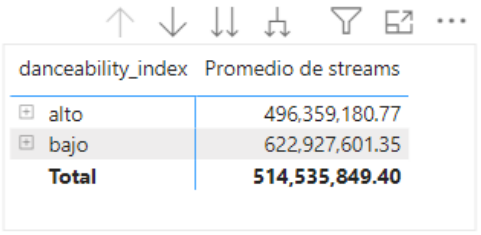
Al momento de leer las teorías, notamos que nos faltaron agregar dos variables para poder resolver la teoría 4. Habíamos hecho anteriormente una tabla temporal pero ahora queremos agregar las columnas. Por ende procedimos a hacer la modificación en la view de tabla completa y agregar las variables faltantes:

```
WITH  total_canciones_por_artista AS (
    SELECT
        artists_name_limpio,
        COUNT(track_id) AS  total_canciones_por_artista
    FROM
        `proyecto-2-hipotesis-426800.spotify_2023.view_ts`
    GROUP BY
        artists_name_limpio
),
total_streams_por_artista AS (
    SELECT
        artists_name_limpio,
        SUM(streams) AS total_streams_por_artista
    FROM
        `proyecto-2-hipotesis-426800.spotify_2023.view_ts`
    GROUP BY
        artists_name_limpio
)
SELECT
    TS.track_id,
    TS.*,
        TC.*,
        TT.*,
    TCA. total_canciones_por_artista,
    TSA.total_streams_por_artista

FROM
    `proyecto-2-hipotesis-426800.spotify_2023.view_ts` AS TS
```

```
LEFT JOIN
    `proyecto-2-hipotesis-426800.spotify_2023.view_tc` AS TC
USING (track_id)
LEFT JOIN
    `proyecto-2-hipotesis-426800.spotify_2023.view_tt` AS TT
USING (track_id)
LEFT JOIN
    total_canciones_por_artista AS TCA
ON
    TS.artists_name_limpio = TCA.artists_name_limpio
LEFT JOIN
    total_streams_por_artista AS TSA
ON
    TS.artists_name_limpio = TSA.artists_name_limpio
```

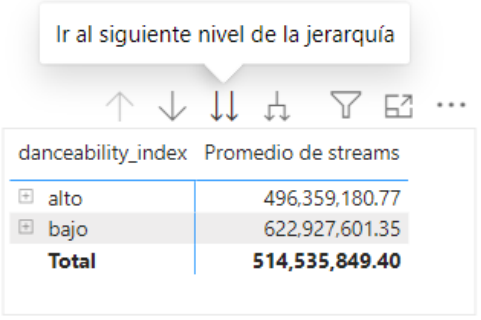
1) Aplicar segmentación.

Creamos una tabla donde se pueda ver el promedio de streams por categoría (alto y bajo) de cada característica de la canción (danceability**, valence**, energy**, acousticness**, instrumentalness**, liveness**, speechiness__):



danceability_index	Promedio de streams
alto	496,359,180.77
bajo	622,927,601.35
Total	**514,535,849.40**

en esta tabla única tenemos todas nuestras características de la canción con su promedio de streams.



danceability_index	Promedio de streams
alto	496,359,180.77
bajo	622,927,601.35
Total	**514,535,849.40**

cuando seleccionamos la dobla flecha hacia abajo, podemos ver las demás características:

valence_index	Promedio de streams
alto	492,221,872.06
bajo	537,960,695.74
Total	**514,535,849.40**

2) Validar hipótesis.

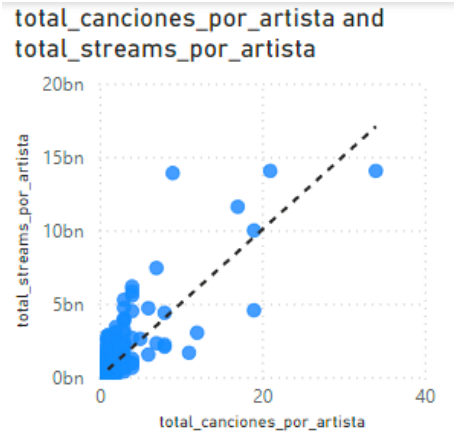
Anteriormente realizamos la correlación de varias variables y colocamos un gráfico scatter plot para cada una explicando la respuesta junto con la hipótesis. Sin embargo nos faltó responder la teoría 4 ya que nos faltaban variables. Aquí la presentamos:

- Correlación total canciones (spotify) por artista vs total streams por artista

  Hipótesis 4:
  ”Los artistas con un mayor número de canciones en Spotify tienen más streams”

```
SELECT
  CORR(total_canciones_por_artista, streams) AS cor_total_canciones_streams
FROM
  `mjproyecto2.database.tabla_full_2`
```

Row	correl_artist_streams
1	0.8907212478807...



El coeficiente de correlación, **0.89...**, confirma esta relación positiva fuerte. Un coeficiente de correlación cercano a 1 indica una correlación positiva muy fuerte.

Existe una relación directa entre la cantidad de canciones que un artista tiene en Spotify y la cantidad de streams que recibe. Cuanto mayor sea el catálogo de un artista en la plataforma, mayor será la probabilidad de que acumule un mayor número de streams.

💡 **ANÁLISIS CON DATOS DE 2023**

Creamos una nueva view donde tendrá solo canciones del año 2023:

```
CREATE VIEW `proyecto-2-hipotesis-426800.spotify_2023.canciones_año_2023` AS
SELECT
  track_id,
  track_name_limpio,
  artists_name_limpio,
  total_canciones_por_artista,
  artist_count,
  total_streams_por_artista,
  released_year,
  released_month,
  released_day,
  fecha_completa,
  tiempo_en_el_mercado,
  in_spotify_playlists,
  in_apple_playlists,
  in_deezer_playlists,
  total_en_playlists,
  in_spotify_charts,
  in_apple_charts,
  in_deezer_charts,
  in_shazam_charts,
  total_en_charts,
  streams,
  categoria_streams,
  bpm,
  categoria_bpm,
  `danceability_%`,
  danceability_index,
  `valence_%`,
  valence_index,
  `energy_%`,
  energy_index,
  `acousticness_%`,
  acousticness_index,
  `instrumentalness_%`,
  instrumentalness_index,
  `liveness_%`,
  liveness_index,
  `speechiness_%`,
  speechiness_index
```

```
FROM
  `proyecto-2-hipotesis-426800.spotify_2023.cuartiles`
WHERE
  released_year = 2023;
```

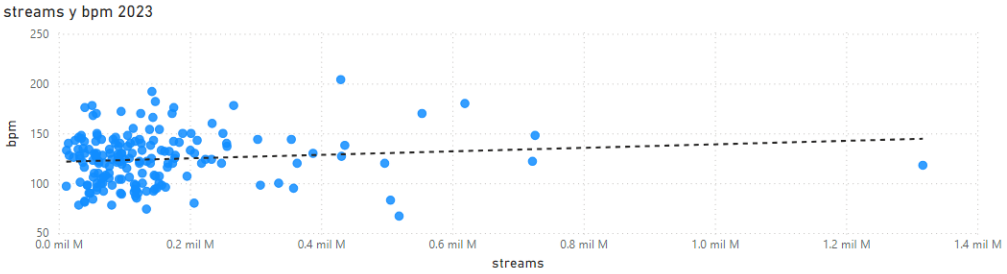
la importaremos a nuestra power BI y calcularemos las correlaciones nuevamente.

Ahora aplicaremos las teorías anteriores solo con canciones el año 2023.

- **Hipótesis 1:** "Las canciones con un mayor BPM (Beats Por Minuto) tienen más éxito en términos de streams en Spotify"

  Correlación streams vs bpm

cor_streams_bpm
0.105271274804...

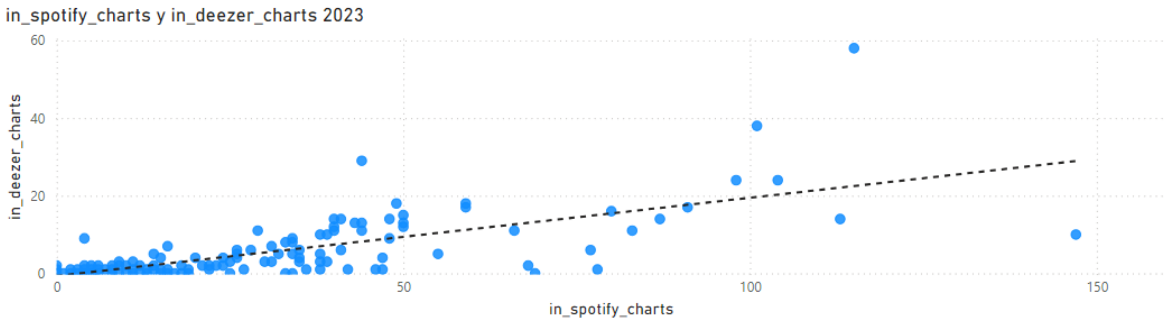


  La correlación de 0.1053 entre BPM y streams de canciones en Spotify indica una relación positiva muy débil. Esto significa que a medida que aumenta el BPM, los streams también tienden a aumentar ligeramente, pero este efecto es muy pequeño. En resumen, el BPM no parece ser un factor significativo en el éxito de una canción en términos de streams.

- **Hipótesis 2:** "Las canciones más populares en el ranking de Spotify también tienen un comportamiento similar en otras plataformas como Deezer"

Correlación in_spotify_charts vs in_deezer_charts:

cor_spotify_deezer_(
0.729599068365...

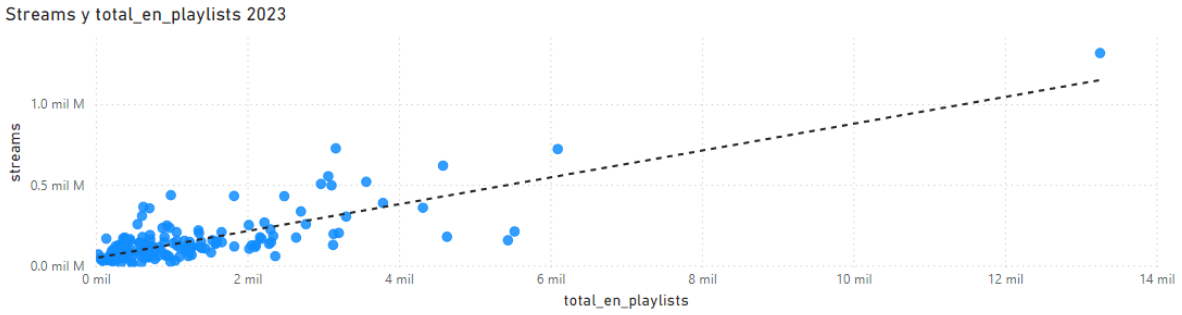


La correlación de 0.7296 entre las posiciones en los charts de Spotify y Deezer indica una relación positiva fuerte. Esto significa que las canciones que son populares en los rankings de Spotify tienden a ser también populares en los rankings de Deezer. En resumen, hay una fuerte coincidencia en el comportamiento de las canciones en ambas plataformas.

- **Hipótesis 3:** "La presencia de una canción en un mayor número de playlists se relaciona con un mayor número de streams"

Correlación total_en_playlists vs streams:

cor_playlists_stream
0.784606122311...

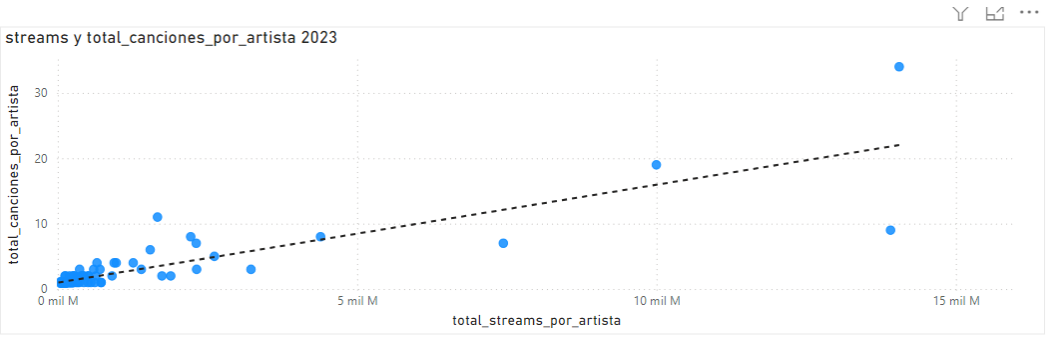


La correlación de 0.7846 entre el número de playlists en las que una canción está presente y el número de streams indica una relación positiva muy fuerte. Esto significa que, en general, las canciones que aparecen en más playlists tienden a tener un mayor número de streams. En resumen, estar presente en más playlists es un factor significativo para el éxito de una canción en términos de streams.

- **Hipótesis 4:** "Los artistas con un mayor número de canciones en Spotify tienen más streams"

Correlación total_canciones_por_artista vs streams:

cor_total_canciones
0.892829742939...

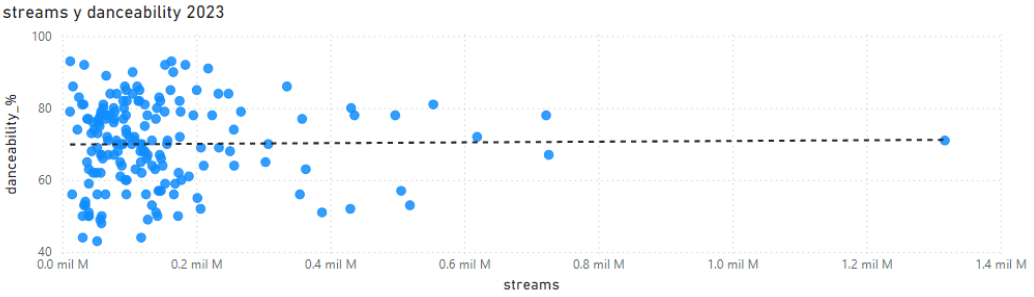


La correlación de 0.8928 entre el número total de canciones por artista y el total de streams por artista indica una relación positiva muy fuerte. Esto significa que, en general, los artistas con más canciones en Spotify tienden a tener un mayor número total de streams. En resumen, tener más canciones en Spotify es un factor significativo para el éxito en términos de streams totales.

- Hipótesis 5: “Las características de la canción influyen en el éxito en términos de streams en Spotify”

- Correlación streams vs danceability_%:

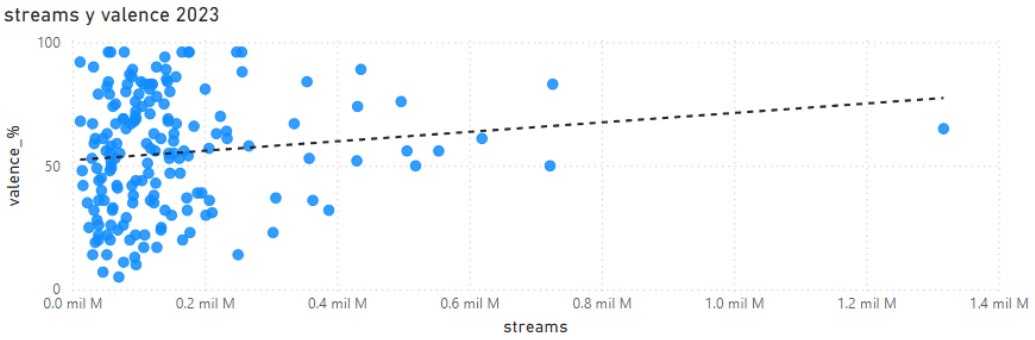
cor_streams_dancea
0.013107170820...



La correlación de 0.0131 entre streams y danceability% indica una relación casi inexistente. Esto significa que la capacidad de una canción para ser bailable (danceability%) no tiene un impacto significativo en el número de streams que recibe. En resumen, el danceability% no parece ser un factor relevante para el éxito de una canción en términos de streams.

- Correlación streams vs valence_%:

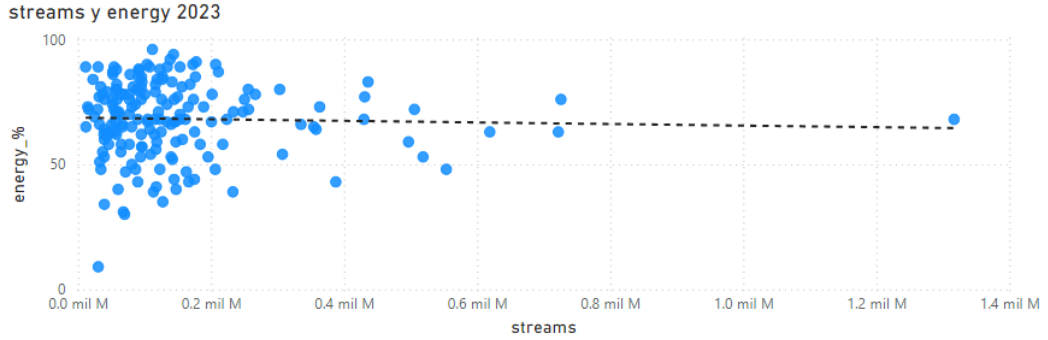
cor_streams_valence
0.124024028350...



La correlación de 0.1240 entre streams y valence% indica una relación positiva pero muy débil. Esto significa que, aunque hay una ligera tendencia de que canciones con un mayor valence% (más positivas, alegres, o eufóricas) tengan más streams, este efecto es mínimo. En resumen, el valence% no es un factor significativo para el éxito de una canción en términos de streams.

- Correlación streams vs energy_%:

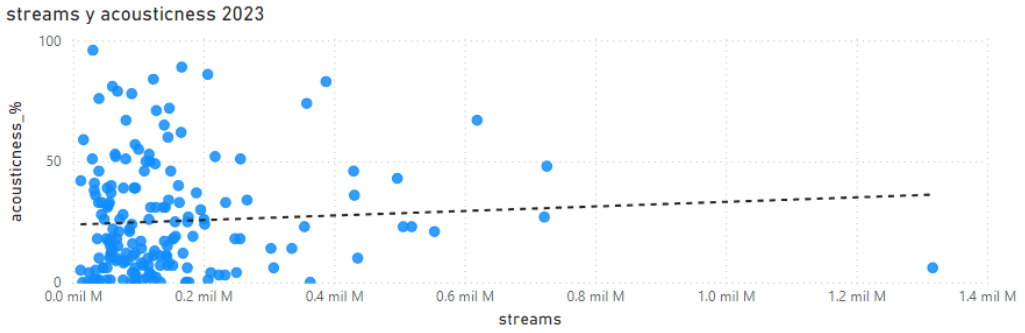
cor_streams_energy
-0.03245045961...



La correlación de -0.0325 entre streams y energy% indica una relación negativa muy débil. Esto significa que la energía de una canción (energy%) tiene un efecto prácticamente nulo y ligeramente inverso sobre el número de streams. En resumen, la energía% no parece ser un factor relevante para el éxito de una canción en términos de streams.

- Correlación streams vs acousticness_%:

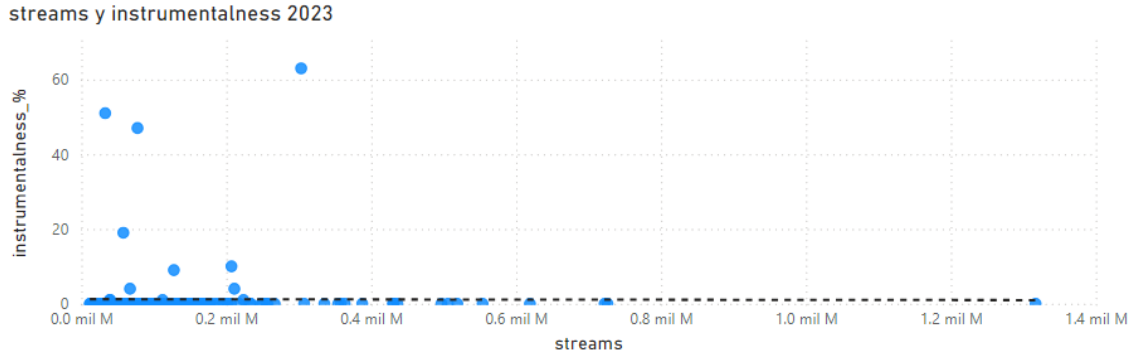
cor_streams_acousti
0.063171658016...



La correlación de 0.0632 entre streams y acousticness% indica una relación positiva muy débil. Esto significa que el nivel de sonidos acústicos en una canción (acousticness%) tiene un efecto mínimo sobre el número de streams. En resumen, el acousticness% no es un factor significativo para el éxito de una canción en términos de streams.

- Correlación streams vs instrumentalness_%:

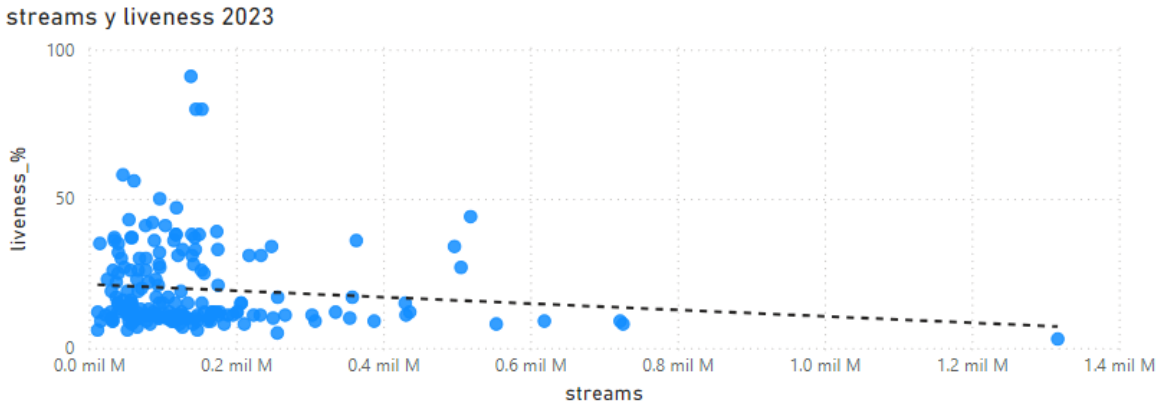
cor_streams_instrum
-0.00443662609...



La correlación de -0.0044 entre streams y instrumentalness% indica una relación negativa extremadamente débil, prácticamente inexistente. Esto significa que el porcentaje de una canción que es instrumental (instrumentalness%) no tiene un impacto significativo en el número de streams. En resumen, el instrumentalness% no es un factor relevante para el éxito de una canción en términos de streams.

- Correlación streams vs liveness_%:

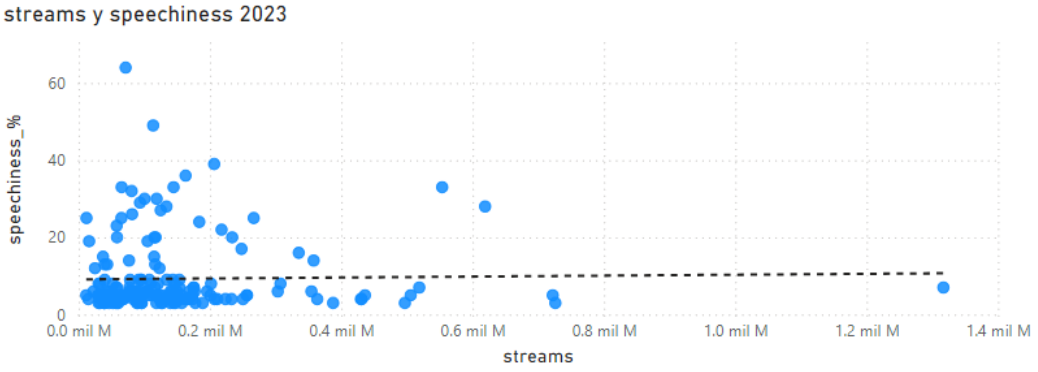
cor_streams_livenes
-0.11590991482...



La correlación de -0.1159 entre streams y liveness% indica una relación negativa débil. Esto significa que a medida que aumenta el porcentaje de liveness (la presencia de elementos que hacen que la canción suene como una actuación en vivo), el número de streams tiende a disminuir ligeramente. En resumen, el liveness% tiene un efecto pequeño y negativo en el éxito de una canción en términos de streams.

- Correlación streams vs speechiness_%:

cor_streams_speech
0.019308880104...



La correlación de 0.0193 entre streams y speechiness% indica una relación positiva extremadamente débil. Esto significa que el porcentaje de palabras habladas en una canción (speechiness%) tiene un impacto insignificante en el número de streams. En resumen, el speechiness% no es un factor relevante para el éxito de una canción en términos de streams.