Ficha Técnica - Projeto 02

Título do Projeto: Hipóteses

Objetivo: O objetivo deste projeto é analisar dados do *Dataset* sobre as *músicas mais ouvidas em 2023* para entender **quais fatores influenciam diretamente o sucesso de uma música**, medido principalmente pelo número de streams.

A gravadora, ao se preparar para lançar um novo artista no mercado global, levantou uma série de hipóteses sobre os elementos que podem estar relacionados ao desempenho de uma música nas plataformas digitais. Entre essas hipóteses estão o impacto do BPM, o número de playlists em que a faixa aparece, o comportamento em outras plataformas como a Deezer, a quantidade de músicas publicadas por um artista, e as características técnicas das faixas.

Este projeto tem como **foco** *refutar* ou *confirmar* essas hipóteses por meio de uma análise de dados rigorosa, utilizando técnicas estatísticas e exploratórias. A partir dos resultados obtidos, serão geradas recomendações estratégicas baseadas em evidências, que ajudarão a gravadora a tomar decisões mais assertivas para maximizar as chances de sucesso do novo artista no cenário musical competitivo e em constante transformação.

Equipe: Gabriela Ferreira Genangelo e Camila Maria de Oliveira Lima Bruschetta.

Ferramentas, Linguagens e Tecnologias:

1. Python
2. SQL
3. Pandas
4. Matplotlib / Seaborn
5. Power BI (DAX)
6. BigQuery
7. Google Colab
8. Jupyter Notebook
9. OpenAI (ChatGPT, API)
10. Loom

Processamento e Análises:

5.1.1 🔵

Após o download do dataset, o arquivo zip foi descompactado, e os trabalhos foram iniciados no BigQuery.
Foi realizada a criação do Projeto, no qual foi nomeado de: “projeto2-hipoteses”.
Depois em ‘Criar Conjunto de Dados’ foi criado o dataset: “spotify_2023”.
Já no dataset “spotify_2023” -> “Criar Tabela”

De início foi tentado o upload do arquivo CSV: *track_in_spotify - spotify*, mas, a importação deu erro, e conforme foi mencionado no erro foi por conta de possivelmente conter caracteres. Criei uma cópia desse arquivo no drive para investigar o que poderia ser, na coluna C em ‘artist(s)_name’ foi localizado o caractere “()”.
Como o BigQuery não permite caracteres especiais como parênteses, colchetes, acentos, símbolos de moeda e a intenção era importar o arquivo direto do computador, de início foi pensado em utilizar o python para a correção, mas se tratava apenas de um dado e o tempo era curto, neste caso foi colocado para editá-lo utilizando o bloco de notas. O caractere foi removido de forma manual e o arquivo foi salvo.
Retornando ao BigQuery foi realizado o mesmo procedimento de antes para importar:

1. Em Origem - Criar Tabela de: Selecionei Fazer upload
2. Em Procurar: Selecionei o arquivo do computador que seria importado
3. Em Destino - Tabela: De início, utilizei o mesmo nome do arquivo de origem: track_in_spotify - spotify
4. Em Esquema - Selecionei: Detectar Automaticamente
5. E Cliquei em: “Criar Tabela”

O arquivo foi importado, e fiz o mesmo procedimento com o segundo arquivo: track_in_competition - competition, pré visualizei a cópia do arquivo no google sheets antes de importar e como não tinha nenhum impedimento, fiz a importação do arquivo seguindo os mesmos passos que eu havia feito acima. A idéia de seguir essa ordem de importação era que as tabelas ficassem na mesma ordem da apresentação da guia, mas, no momento em que o arquivo foi importado descobri que o BigQuery exibe as tabelas sempre em ordem alfabética e não na ordem importada, sabendo que não há como renomear tabelas no BigQuery e que a ordenação no console segue a ordem

alfabética, optei por excluir ambas as tabelas e refazer o procedimento de importação, adicionando um número antes do nome de cada tabela para garantir a ordenação desejada, e aproveitei também para, desta vez, salvar os nomes sem utilizar espaços ou o caractere “-” (hífen), imaginando que manter esse formato poderia causar problemas de identificação no BigQuery em alguma das etapas futuras.

No 3° arquivo: track_technical_info - technical_info na hora de pré analisar a cópia no google sheets, sete colunas continham caracteres de “%” cheguei a testar a importação estando dessa forma, diferente do “()”, os dados mesmo estando com “%” foram importados para o BigQuery, porém, considerando que em algum momento isso possa causar problemas, fiz o tratamento dos dados novamente usando o bloco de notas. Removi todos os “%” dos cabeçalhos, e escrevi “pct” no lugar do caractere, salvei o arquivo e repeti o mesmo procedimento de antes para importar para o BigQuery.

O dataset “‘spotify_2023” no BigQuery ficou com as seguintes tabelas, organizadas nessa ordem:
1track_in_spotify
2track_in_competition
3track_technical_info

5.1.2 🔵

Na parte de identificar e tratar os nulos, para contar o número de registros onde cada uma das colunas continham dados nulos e vazios e para contar também o valor total de registros na tabela. Iniciei formatando a Query, e para os dados “track_in_spotify” utilizei:

```
SELECT
COUNTIF(track_id IS NULL OR TRIM(track_id) = '') AS nulos_ou_vazios_track_id,
COUNTIF(track_name IS NULL OR TRIM(track_name) = '') AS
nulos_ou_vazios_track_name,
COUNTIF(artist_name IS NULL OR TRIM(artist_name) = '') AS
nulos_ou_vazios_artist_name,
COUNTIF(artist_count IS NULL OR TRIM(CAST(artist_count AS STRING)) = '') AS
nulos_ou_vazios_artist_count,
COUNTIF(released_year IS NULL OR TRIM(CAST(released_year AS STRING)) = '') AS
nulos_ou_vazios_released_year,
COUNTIF(released_month IS NULL OR TRIM(CAST(released_month AS STRING)) = '')
AS nulos_ou_vazios_released_month,
```

```
COUNTIF(released_day IS NULL OR TRIM(CAST(released_day AS STRING)) = '') AS
nulos_ou_vazios_released_day,
COUNTIF(in_spotify_playlists IS NULL OR TRIM(CAST(in_spotify_playlists AS
STRING)) = '') AS nulos_ou_vazios_in_spotify_playlists,
COUNTIF(in_spotify_charts IS NULL OR TRIM(CAST(in_spotify_charts AS STRING)) =
'') AS nulos_ou_vazios_in_spotify_charts,
COUNTIF(streams IS NULL OR TRIM(CAST(streams AS STRING)) = '') AS
nulos_ou_vazios_streams,
COUNT(*) AS total_registros
FROM `projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`
```

Com isso foi possível identificar 2 nulos na coluna “track_name”.
Salvei essa query como: limpeza_dados_1spotify e deixei com visibilidade público para que qualquer pessoa com o link possa consultar.
Para tratar estes 2 nulos encontrados, com o intuito de manter o registro sem perder os demais dados relevantes para a análise, em: “limpeza_dados_1spotify “, utilizei a query:

```
SELECT
IF(TRIM(track_name) IS NULL OR TRIM(track_name) = '', 'Track Name Não
Informada', track_name) AS track_name_null_corrigido,
  *
FROM `projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`
```

A query criou uma nova variável chamada track_name_null_corrigido, na qual os valores nulos ou vazios da coluna track_name foram imputados com o texto “Track Name Não Informada”.

Para verificar os nulos e vazios dos dados “**track_in_competition**” foi utilizado:

```
SELECT
COUNTIF(track_id IS NULL OR TRIM(track_id) = '') AS nulos_ou_vazios_track_id,
COUNTIF(in_apple_playlists IS NULL OR TRIM(CAST(in_apple_playlists AS STRING))
= '') AS nulos_ou_vazios_in_apple_playlists,
COUNTIF(in_apple_charts IS NULL OR TRIM(CAST(in_apple_charts AS STRING)) = '')
AS nulos_ou_vazios_in_apple_charts,
COUNTIF(in_deezer_playlists IS NULL OR TRIM(CAST(in_deezer_playlists AS
STRING)) = '') AS nulos_ou_vazios_in_deezer_playlists,
COUNTIF(in_deezer_charts IS NULL OR TRIM(CAST(in_deezer_charts AS STRING)) =
'') AS nulos_ou_vazios_in_deezer_charts,
COUNTIF(in_shazam_charts IS NULL OR TRIM(CAST(in_shazam_charts AS STRING)) =
'') AS nulos_ou_vazios_in_shazam_charts,
COUNT(*) AS total_registros
FROM `projeto2-hipoteses-459302.spotify_2023.2track_in_competition`
```

Com essa query foi possível identificar 50 nulos na coluna “in_shazam_charts”.
Salvei essa query como: limpeza_dados_2competition e também deixei com visibilidade público.
Para os 50 nulos encontrados, na Query “limpeza_dados_2competition”
Realizei uma análise dos dados, utilizando:

```
SELECT
*
FROM `projeto2-hipoteses-459302.spotify_2023.2track_in_competition`
```

Ao analisar a coluna referente à presença e classificação da música nas paradas da Shazam, observei que os valores 0 indicam que a música não foi classificada, o que considero um dado válido. Já os valores NULL me fizeram refletir sobre a possibilidade de terem ocorrido por falhas na coleta, dados corrompidos ou ausência de informação.

Cheguei a considerar a imputação do valor -1 para representar essa ausência de dados, com a ideia de tratá-lo posteriormente durante os cálculos, desconsiderando esse valor nas métricas. No entanto, entendi que essa abordagem poderia gerar confusão, já que o -1 é um valor numérico e, se não for tratado corretamente em todas as etapas, pode afetar negativamente médias, somas e outras análises estatísticas, comprometendo a consistência dos resultados.

Por esse motivo, de inicio optei por manter os valores como NULL, pois entendo que eles representam de forma fiel a inexistência de dados confiáveis, enquanto o 0 já cumpre o papel de indicar que a música não foi classificada.

Como antes eu havia feito a query considerando nulos e vazios juntos, a fim de ter certeza que todos os 50 se tratavam mesmo de NULL, testei as consultas abaixo separadamente:

Para contar vazio, utilizei a query:

```
SELECT
COUNTIF(TRIM(CAST(in_shazam_charts AS STRING)) = '') AS
vazios_in_shazam_charts
FROM `projeto2-hipoteses-459302.spotify_2023.2track_in_competition`
```

E os dados retornaram sem nenhuma informação.

Para contar nulos, utilizei:

```
SELECT
COUNTIF(in_shazam_charts IS NULL) AS nulos_in_shazam_charts
FROM `projeto2-hipoteses-459302.spotify_2023.2track_in_competition`
```

Confirmando que os 50 se tratavam mesmo de “null”

Neste caso, conforme as informações que apresentei acima, e já que o BigQuery ignora os valores “null”, de início eu mantive estes 50 nulos.

Para os dados “track_technical_info” seguindo os mesmos passos dos anteriores, utilizei a query:

```
SELECT
COUNTIF(track_id IS NULL OR TRIM(track_id) = '') AS nulos_ou_vazios_track_id,
COUNTIF(bpm IS NULL OR TRIM(CAST(bpm AS STRING)) = '') AS nulos_ou_vazios_bpm,
COUNTIF(`key` IS NULL OR TRIM(CAST(`key` AS STRING)) = '') AS
nulos_ou_vazios_key,
COUNTIF(mode IS NULL OR TRIM(CAST(mode AS STRING)) = '') AS
nulos_ou_vazios_mode,
COUNTIF(`danceability_pct` IS NULL OR TRIM(CAST(`danceability_pct` AS STRING))
= '') AS nulos_ou_vazios_danceability,
COUNTIF(`valence_pct` IS NULL OR TRIM(CAST(`valence_pct` AS STRING)) = '') AS
nulos_ou_vazios_valence,
COUNTIF(`energy_pct` IS NULL OR TRIM(CAST(`energy_pct` AS STRING)) = '') AS
nulos_ou_vazios_energy,
COUNTIF(`acousticness_pct` IS NULL OR TRIM(CAST(`acousticness_pct` AS STRING))
= '') AS nulos_ou_vazios_acousticness,
COUNTIF(`instrumentalness_pct` IS NULL OR TRIM(CAST(`instrumentalness_pct` AS
STRING)) = '') AS nulos_ou_vazios_instrumentalness,
COUNTIF(`liveness_pct` IS NULL OR TRIM(CAST(`liveness_pct` AS STRING)) = '')
AS nulos_ou_vazios_liveness,
COUNTIF(`speechiness_pct` IS NULL OR TRIM(CAST(`speechiness_pct` AS STRING)) =
'') AS nulos_ou_vazios_speechiness,
COUNT(*) AS total_registros
FROM `projeto2-hipoteses-459302.spotify_2023.3track_technical_info`
```

Foram identificados 95 valores nulos na coluna “Key".

A Query foi salva como: “limpeza_dados_3technical” com visibilidade publico.

Para estes 95 dados encontrados, realizei a mesma análise que fiz com os dados anteriores, utilizando a query:

```
SELECT

*

FROM `projeto2-hipoteses-459302.spotify_2023.3track_technical_info`
```

Observei que os dados encontrados estavam classificados mesmo como null, como essa coluna representa o **tom musical da música**, que é uma **característica musical** da faixa,e ela está relacionada diretamente com a **última hipótese** levantada: “As características da música influenciam o sucesso em termos de número de streams no Spotify.” mesmo tendo outros dados que futuramente poderiam me ajudar a refutar ou confirmar, descartar essa informação poderia afetar a validade das análises relacionadas a essa hipótese.

Para confirmar que todos os 95 dados encontrados se tratava de “null” utilizei as seguintes consultas:

Para contar vazio, utilizei:

```
SELECT

COUNTIF(TRIM(CAST(`key` AS STRING)) = '') AS vazios_key

FROM `projeto2-hipoteses-459302.spotify_2023.3track_technical_info`
```

Que retornou sem nenhuma informação validada.

E para contar os nulos, utilizei essa:

```
SELECT

COUNTIF(`key` IS NULL) AS nulos_key

FROM `projeto2-hipoteses-459302.spotify_2023.3track_technical_info`
```

Que confirmou a presença dos 95 registros nulos, validando que todos realmente estavam ausentes de informação.

Para tratar este caso, considerando que o BigQuery ignora valores nulos em cálculos, de início optei por manter a coluna “key” da forma que ela estava, e criar uma nova variável com uma string, imputando a informação “Sem Tom Informado”. Essa abordagem me permite utilizar essa nova variável em visualizações gráficas futuras, ciente de que ela será adequada apenas para análises descritivas ou categóricas, e não para cálculos numéricos, para estes casos eu terei que utilizar a variável “key”.

A consulta utilizada para criar a nova variável “key_musical_label” foi:

```
SELECT
```

```
track_id,

bpm,

`key`,

mode,

`danceability_pct`,

`valence_pct`,

`energy_pct`,

`acousticness_pct`,

`instrumentalness_pct`,

`liveness_pct`,

`speechiness_pct`,

CASE

WHEN `key` IS NULL OR TRIM(CAST(`key` AS STRING)) = '' THEN 'Sem Tom
Informado'

ELSE CAST(`key` AS STRING)

END AS key_musical_label

FROM `projeto2-hipoteses-459302.spotify_2023.3track_technical_info`
```

Dessa forma, nessa etapa, segui mantendo os dados nulos originais na coluna “key”, e criei uma versão transformada da coluna, que substitui os valores nulos ou vazios por "Sem Tom Informado", facilitando a análise descritiva e categórica em futuras visualizações gráficas, sem impactar os cálculos que dependem da coluna original.

5.1.3 🔵

Para identificar dados duplicados, primeiro verifiquei de forma mais simples, considerando todas as variáveis existentes.

Para os dados de “track_in_spotify” foi utilizado a query:

```
SELECT
```

```
track_id,

track_name,

artist_name,

artist_count,

released_year,

released_month,

released_day,

in_spotify_playlists,

in_spotify_charts,

streams,

COUNT(*) AS total_duplicadas

FROM

`projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`

GROUP BY

track_id,

track_name,

artist_name,

artist_count,

released_year,

released_month,

released_day,

in_spotify_playlists,

in_spotify_charts,

streams

HAVING
```

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

A consulta não retornou com nenhuma duplicidade.

Utilizei a mesma consulta desconsiderando “track_name” que foi onde eu havia encontrado os dados nulos, com o objetivo de **evitar falsos negativos na identificação de registros duplicados**, já que campos ausentes (nulos ou em branco) podem variar entre linhas que, na prática, representam o mesmo item.

A query me retornou sem nenhuma duplicidade.

Também verifiquei se os dois artistas que apareciam sem o nome da música poderiam estar vinculados a outros dados na base, com o intuito de identificar possíveis duplicidades. No entanto, para ambos os casos, havia apenas um único registro do artista na tabela.

Quando verifiquei duplicidades entre “track_name” e “artist_name” utilizando:

```
WITH duplicadas AS (

SELECT

track_name,

artist_name

FROM

`projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`

GROUP BY

track_name,

artist_name

HAVING

COUNT(*) > 1

)

SELECT

t.track_id,

t.track_name,
```

```
t.artist_name,

t.artist_count,

t.released_year,

t.released_month,

t.released_day,

t.in_spotify_playlists,

t.in_spotify_charts,

t.streams

FROM

`projeto2-hipoteses-459302.spotify_2023.1track_in_spotify` t

JOIN

duplicadas d

ON

t.track_name = d.track_name AND

t.artist_name = d.artist_name
```

Foram identificadas 4 pares de registros duplicados, ou seja, 8 registros no total, dos seguintes "`track_id`": 5080031, 7173596, 5675634, 3814670, 4967469, 8173823, 1119309, 4586215.

Embora os registros duplicados apresentassem algumas variações em determinadas variáveis, tratavam-se da mesma música e do mesmo artista. Como não foi possível identificar com clareza a origem dessas inconsistências nos dados, optei por nos próximos passos, removê-las do conjunto.

Para verificar as duplicidades da tabela "**track_in_competition" utilizei a query:**

```
SELECT *

FROM `projeto2-hipoteses-459302.spotify_2023.2track_in_competition`;

SELECT

track_id,
```

```
in_apple_playlists,

in_apple_charts,

in_deezer_playlists,

in_deezer_charts,

in_shazam_charts,

COUNT(*) AS total_duplicados

FROM

`projeto2-hipoteses-459302.spotify_2023.2track_in_competition`

GROUP BY

track_id,

in_apple_playlists,

in_apple_charts,

in_deezer_playlists,

in_deezer_charts,

in_shazam_charts

HAVING

COUNT(*) > 1
```

A query retornou os dados sem nenhuma duplicidade. Seguindo o mesmo raciocínio da tabela anterior, a consulta foi refeita desconsiderando a coluna que possuía os nulos “in_shazam_charts” e não foram identificados dados duplicados, porém, como foram encontrados 8 registros duplicados na tabela anterior, os quais decidi que irei remover, os dados dos respectivos track_id também serão excluídos desta tabela, garantindo a consistência entre as bases.

Para a tabela “3track_technical_info” a query utilizada foi:

```
SELECT

track_id,

bpm,
```

```
`key`,

mode,

`danceability_pct`,

`valence_pct`,

`energy_pct`,

`acousticness_pct`,

`instrumentalness_pct`,

`liveness_pct`,

`speechiness_pct`,

CASE

WHEN `key` IS NULL OR TRIM(CAST(`key` AS STRING)) = '' THEN 'Sem Tom
Informado'

ELSE CAST(`key` AS STRING)

END AS key_musical_label

FROM `projeto2-hipoteses-459302.spotify_2023.3track_technical_info`;



SELECT

track_id,

bpm,

key,

mode,

danceability_pct,

valence_pct,

energy_pct,

acousticness_pct,

instrumentalness_pct,
```

```
liveness_pct,

speechiness_pct,

COUNT(*) AS total_duplicados

FROM

`projeto2-hipoteses-459302.spotify_2023.3track_technical_info`

GROUP BY

track_id,

bpm,

key,

mode,

danceability_pct,

valence_pct,

energy_pct,

acousticness_pct,

instrumentalness_pct,

liveness_pct,

speechiness_pct

HAVING

COUNT(*) > 1
```

Sem nenhuma duplicidade encontrada, também verifiquei desconsiderando a coluna que foi identificado os dados nulos “key”, e nenhuma outra duplicidade foi encontrada.

*Durante o processo de unificação das tabelas, os 8 `track_id` identificados como duplicados na primeira tabela,”1track_in_spotify”, serão removidos de todas as tabelas envolvidas.

5.1.4 🔵

Após revisar as colunas que continham grandes quantidades de dados nulos, “**key e** “**in_shazam_charts”**, e analisar sua relação com as hipóteses levantadas, nessa etapa do projeto decidi removê-las da análise.

A variável **key** (tom musical) apresentou **baixo valor informativo** quando comparada a outras variáveis, como **bpm**, **danceability_pct**, **energy_pct**, e **speechiness_pct**, que têm um impacto mais claro nas hipóteses relacionadas ao comportamento do ouvinte e ao desempenho nas plataformas de streaming. Além disso, a alta **quantidade de dados nulos** poderia comprometer a qualidade da análise.

Feito isso também optei por não incluir na análise a variável que eu havia criado “key_musical_label”.

Já a variável “**in_shazam_charts”** não possui dados complementares que permitam uma análise integrada como as variáveis de **Spotify**, **Deezer** e **Apple Music**, que oferecem uma visão mais completa do desempenho da música nas paradas e playlists. A **falta de dados complementares** e a **quantidade considerável de dados nulos** me fizeram optar também pela remoção dessa variável.

Foi utilizado essa query na tabela “2track_in_competition”:

```
SELECT

* EXCEPT(in_shazam_charts)

FROM

`projeto2-hipoteses-459302.spotify_2023.2track_in_competition`
```

E essa, na tabela “3track_technical_info”:

```
SELECT

* EXCEPT(key)

FROM

`projeto2-hipoteses-459302.spotify_2023.3track_technical_info`
```

5.1.5 🔵

Com o intuito de identificar registros da tabela “1track_in_spotify” em que as colunas “track_name” e “artist_name” continham caracteres especiais, símbolos ou pontuações que poderiam interferir em análises padronizadas, foi realizado a seguinte consulta para localizar qualquer caractere que não seja letra (a-z, A-Z), número (0-9) ou espaço:

```
SELECT

*

FROM

`projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`

WHERE

REGEXP_CONTAINS(track_name, r'[^a-zA-Z0-9\s]') OR

REGEXP_CONTAINS(artist_name, r'[^a-zA-Z0-9\s]');
```

Foram aplicadas técnicas de limpeza de dados nos campos track_name e artist_name, com o objetivo de padronizar a formatação textual. O processo incluiu a conversão de letras para minúsculas, remoção de caracteres especiais, normalização de espaços em branco e eliminação de espaços desnecessários. Para isso, foi utilizada a query:

```
SELECT

*,

REGEXP_REPLACE(

REGEXP_REPLACE(

LOWER(track_name),

r'[^a-z0-9\s]', ''),

r'\s+', ' '

) AS track_name_cleaned,


REGEXP_REPLACE(
```

```
REGEXP_REPLACE(

LOWER(artist_name),

r'[^a-z0-9\s]', ''),

r'\s+', ' '

) AS artist_name_cleaned



FROM

`projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`
```

Aproveitando a função REGEXP, verifiquei se existia strings fora do padrão a fim de identificá-las e corrigi las, utilizei a função nas 3 tabelas verificando a coluna chave “Track_id”, e nas três tabelas foi identificado um track_id neste formato: **0:00.**

A consulta utilizada foi:

```
SELECT track_id

FROM `projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`

WHERE REGEXP_CONTAINS(CAST(track_id AS STRING), r'[^a-zA-Z0-9]')

OR NOT REGEXP_CONTAINS(CAST(track_id AS STRING), r'^\d{7}$')
```

E para as outras duas tabelas segui o mesmo formato, modificando apenas o “From”.

Como em todas as tabelas havia o mesmo total de linhas, para ter certeza que este track_id se tratava do mesmo, em todas as tabelas, eu utilizei a query:

```
WITH

--- Tabela 1:

t1 AS (

SELECT DISTINCT track_id FROM
`projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`

),

--- Tabela 2:

t2 AS (
```

```sql
SELECT DISTINCT track_id FROM
`projeto2-hipoteses-459302.spotify_2023.2track_in_competition`

),

--- Tabela 3:

t3 AS (

SELECT DISTINCT track_id FROM
`projeto2-hipoteses-459302.spotify_2023.3track_technical_info`

),

--- União de todos os track_id

todos_ids AS (

SELECT track_id FROM t1

UNION DISTINCT

SELECT track_id FROM t2

UNION DISTINCT

SELECT track_id FROM t3

)

--- Consulta final com verificação de presença e filtro

SELECT

track_id,

IF(track_id IN (SELECT track_id FROM t1), 1, 0) AS em_t1,

IF(track_id IN (SELECT track_id FROM t2), 1, 0) AS em_t2,

IF(track_id IN (SELECT track_id FROM t3), 1, 0) AS em_t3

FROM todos_ids

WHERE NOT (track_id IN (SELECT track_id FROM t1)

AND track_id IN (SELECT track_id FROM t2)

AND track_id IN (SELECT track_id FROM t3))
```

```
ORDER BY track_id;
```

Basicamente a consulta verificou a consistência do campo track_id entre as três tabelas diferentes. Com o intuito de identificar track_ids que **não** estariam presentes simultaneamente nas três tabelas. A consulta retornou como “Não há dados para exibir”, confirmando que o track_id 0:00 se tratava da mesma música. Neste caso, eu poderia criar uma sequência de 7 dígitos respeitando o padrão, desde que essa nova sequência ainda não existisse, e inseri-las nas 3 tabelas.

Salvei essa query separadamente como “comparativo_track_id_consistente”.

Para realizar a correção do track_id que estava fora de padrão eu utilizei:

```
WITH todos_ids AS (

SELECT *, CAST(track_id AS STRING) AS track_id_str

FROM `projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`

),

contagem AS (

SELECT COUNT(*) AS total FROM todos_ids

),

possiveis_ids AS (

SELECT

LPAD(CAST(1000000 + OFFSET AS STRING), 7, '0') AS novo_track_id

FROM contagem,

UNNEST(GENERATE_ARRAY(0, total * 2)) AS OFFSET

),

primeiro_id_disponivel AS (

SELECT novo_track_id

FROM possiveis_ids

WHERE novo_track_id NOT IN (SELECT track_id_str FROM todos_ids)

LIMIT 1
```

```
)

SELECT

*,

CASE

WHEN track_id_str = '0:00' THEN (SELECT novo_track_id FROM
primeiro_id_disponivel)

ELSE track_id_str

END AS track_id_corrigido

FROM todos_ids;
```

Realizei o mesmo processo nas demais tabelas, adaptando conforme a estrutura de cada uma. Para garantir a consistência dos dados, criei uma nova coluna em cada tabela, na qual substitui os valores fora do padrão, especificamente o track_id no formato "0:00" por um número inteiro de 7 dígitos (`1000000`). Esse novo valor foi gerado de forma única, assegurando que ele ainda não estivesse presente na coluna original, preservando assim a integridade dos identificadores em todo o conjunto de dados.

Na coluna “**Streams**”, que estava sendo considerada como **STRING**, também foram encontrados dados fora do padrão. No entanto, considerando que essa coluna deveria ser do tipo **INTEGER** e seguindo as orientações da próxima meta da guia, decidi tratá-la mais adiante.

Na tabela “**3track_technical_info**” em “**mode**” não foi localizado nenhuma **string** fora do padrão.

Dessa forma segui para a próxima meta da guia.

5.1.6 🔵

Utilizei as funções MIN, MAX e AVG nos dados do tipo INTEGER.

Na tabela “track_in_spotify”, coluna “released_year” havia anos muito fora da média, além de que fazendo uma análise mais aprofundada encontrei algumas inconsistências nos dados como a música “A Holly Jolly Christmas”, do artista: Burl Ives, no Dataset o lançamento dela estava como se tivesse ocorrido no ano de 1952 mas, pela minha pesquisa, o lançamento ocorreu alguns anos depois. Como é possível que o dataset seja “ilusório” e não eram muitos anos de diferença do lançamento, decidi manter dados como esse, da forma que estavam, exceto a musica “Agudo M gi” dos artistas Styrx, utku INC, Thezth que estava como data de lançamento de 1930, esse eu considerei como outlier, acredito que ocorreu algum erro, o nome dessa musica imagino que deva ser algo como ”Agudo Mágico” e de artistas que são da atualidade, para este caso irei desconsiderar a linha de dados desse track_id “2475712” da análise.

Utilizei:

```
SELECT *

FROM `projeto2-hipoteses-459302.spotify_2023.1track_in_spotify`

WHERE released_year > 1930;
```

Para me trazer apenas os dados onde o lançamento teria ocorrido em data posterior a 1930.

Nas demais variáveis desta tabela não encontrei nada que considerasse relevante na meta.

Na tabela **“2track_in_competition”** em **“in_deezer_playlists”** Identifiquei valores significativamente acima do esperado, o que levantou a hipótese de um possível erro no processo de importação. Para investigar, consultei os dados de origem no Google Sheets e constatei a presença de dados com casas decimais, indicando uma provável falha na interpretação numérica.

Criei a variavel: **“busca_de_casa_decimal”** e com a query: **=SE(REGEXMATCH(TO_TEXT(D2); ","); "Dado casa decimal"; D2)** filtrei os que estavam como “Dado casa Decimal”, foram encontrados no total **79 registros**, o que representava cerca de **8,3%** da base, comparei esses registros com os números discrepantes identificados no BigQuery e percebi que, ao realizar a importação sem a devida verificação prévia, os valores originalmente com casas decimais foram

convertidos incorretamente, resultando na perda dessas casas decimais durante o processo.

Analisei esses dados e fui comparando com a forma como entraram no BigQuery, com o objetivo de corrigi-los, mas, devido à estrutura numérica inconsistente, indicando possível falha na padronização do separador decimal ou inconsistência na origem dos dados e à ausência de uma fonte confiável para validação dos valores corretos, optei pela exclusão desses registros incertos, a fim de preservar a integridade da análise.

A seguir, estão listados os **79 track_id** que foram removidos:

1. *4367298*
2. *2391919*
3. *2553984*
4. *4527090*
5. *8895699*
6. *3126367*
7. *8502696*
8. *8408827*
9. *7159285*
10. *1216337*
11. *2394886*
12. *8797570*
13. *4350729*
14. *8965193*
15. *3605651*
16. *3869741*
17. *4826716*
18. *3253565*
19. *4181228*
20. *1663974*
21. *3412305*
22. *7325433*
23. *1928790*
24. *4242212*
25. *2664577*
26. *8580552*
27. *2031309*
28. *5999455*
29. *4299313*
30. *6316146*
31. *2008299*
32. *7256696*
33. *7451979*

34. 8213022
35. 2670198
36. 6766942
37. 4238748
38. 7587856
39. 6380084
40. 3049716
41. 4552573
42. 1776310
43. 2143818
44. 6891911
45. 1967014
46. 4490744
47. 7224008
48. 2237089
49. 3612400
50. 7456153
51. 2969661
52. 4152167
53. 5615333
54. 1512850
55. 5074676
56. 1486311
57. 2635416
58. 2285737
59. 3333614
60. 4918126
61. 2828618
62. 3363172
63. 5223297
64. 5694303
65. 3181974
66. 1660982
67. 8217918
68. 3739389
69. 7405938
70. 6523480
71. 5831523
72. 3780370
73. 4096612
74. 5906167
75. 5324238
76. 2188178
77. 7080917

78. 8455676
79. 7890756

Após identificar e remover os registros inconsistentes, excluí a variável auxiliar criada para essa análise e salvei a planilha atualizada como um novo arquivo .csv. Em seguida, excluí a tabela correspondente no BigQuery e realizei uma nova importação, seguindo o mesmo procedimento utilizado inicialmente.

Após esse processo notei que o id do projeto mudou, neste caso foi necessário atualizar todas as consultas que eu havia feito utilizando a informação do novo id.

O que antes era: ***projeto2-hipoteses-459302.spotify_2023.2track_in_competition***

passou a ser: ***projeto2-hipoteses-459613.spotify_2023.2track_in_competition***

Realizei uma consulta inicial para verificar os dados importados e observei que o track_id anteriormente exibido como 0:00 passou a ser representado como **00:00** nesta nova importação. Após confirmar que se tratava do mesmo registro, atualizei a query responsável pela identificação e correção.

Importei novamente o arquivo de origem com o objetivo de comparar os dados lado a lado. Após essa verificação, não identifiquei mais nenhum registro com indícios de inconsistência. Diante disso, procedi com a exclusão dessa tabela com os dados de origem, uma vez que eu havia importado apenas para fins de comparação.

Na tabela **“3track_technical_info”** não foi identificado nenhum dado discrepante em variáveis numéricas.

Retornando à tabela **“track_in_spotify”**, na variável “streams”, identifiquei um valor não numérico associado ao ***track_id*** **4061483**. Esse dado inconsistente provavelmente foi o responsável por fazer com que o BigQuery interpretasse automaticamente toda a coluna como **string**, ao invés de **INTEGER**.

Seguindo as orientações da guia, avancei para a próxima meta para continuar o processo de correção.

5.1.7 🔵

Devido à presença de um valor não numérico na variável **"streams"**, associado ao **track_id 4061483,** e considerando que não foi possível recuperar o valor original, optei por **excluir** essa linha do conjunto de dados. Essa decisão se justifica pelo fato de que a variável "streams" é fundamental para a análise e para responder às hipóteses deste projeto. Para realizar essa exclusão, utilizei a seguinte query:

```
SELECT

*,

SAFE_CAST(streams AS INT64) AS streams_corrigidos

FROM `projeto2-hipoteses-459613.spotify_2023.1track_in_spotify`

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

A consulta retornou a coluna “streams” já convertida para o tipo/schema: **INTEGER** em uma *nova variável* nomeada de **“streams_corrigidos”**.

5.1.8 🔵

Para criar uma variável única de data de lançamento, foi utilizada a seguinte query:

```
SELECT

*,

DATE(

CAST(released_year AS INT64),

CAST(released_month AS INT64),

CAST(released_day AS INT64)

) AS release_date

FROM `projeto2-hipoteses-459613.spotify_2023.1track_in_spotify`;
```

A consulta me retornou uma nova variável contendo a data de lançamento no formato padrão *aaaa-mm-dd*, permitindo uma análise temporal consistente.

Em relação a criar uma variável para representar a *participação total nas playlists*, a fim de garantir a integridade dos dados, foi decidido primeiro unificar as tabelas.

5.1.9 🔵

Durante a preparação das *views, antes* da unificação das tabelas, identifiquei que nem todas as hipóteses levantadas pela gravadora fariam o uso da variável

“in_deezer_playlists”. Para aproveitar melhor os dados, optei por criar duas versões da tabela unificada:

1. **Tabela Unificada Filtrada “**tabela_unificada_tracks_filtrada”**(com menos linhas, mas mais variáveis):** que será utilizada nas análises que requerem a variável in_deezer_playlists, como, por exemplo, a hipótese que avalia se músicas populares no Spotify também apresentam comportamento semelhante na Deezer.

2. **Tabela Unificada Expandida** “tabela_unificada_tracks_expandida”**(com mais linhas, porém com menos variáveis):** utilizada nas hipóteses que não dependem da variável in_deezer_playlists. Um exemplo é a hipótese **“Artistas com um maior número de músicas no Spotify têm mais streams”**, que depende apenas de variáveis como o nome do artista, quantidade de músicas e total de streams, e pode ser analisada com um maior número de registros.

Acredito que, ao aplicar essa abordagem, poderei adaptar a base de dados às necessidades de cada análise, garantindo mais qualidade estatística quando necessário, sem perder informações relevantes.

1.Tabela Unificada Filtrada

Para criar a view: **“view_1track_in_spotify_limpa”**, da tabela **“1track_in_spotify”** foi utilizado essa query:

```
CREATE OR REPLACE VIEW
projeto2-hipoteses-459613.spotify_2023.view_1track_in_spotify_limpa AS

WITH pares_contados AS (

SELECT

track_name,

artist_name,

COUNT(*) AS qtd

FROM `projeto2-hipoteses-459613.spotify_2023.1track_in_spotify`

GROUP BY track_name, artist_name

),

pares_unicos AS (
```

```sql
SELECT

track_name,

artist_name

FROM pares_contados

WHERE qtd = 1

),

dados_filtrados AS (

SELECT t.*

FROM `projeto2-hipoteses-459613.spotify_2023.1track_in_spotify` t

JOIN pares_unicos u

ON t.track_name = u.track_name

AND t.artist_name = u.artist_name

),

todos_ids AS (

SELECT *, CAST(track_id AS STRING) AS track_id_str

FROM dados_filtrados

),

contagem AS (

SELECT COUNT(*) AS total FROM todos_ids

),

possiveis_ids AS (

SELECT

LPAD(CAST(1000000 + OFFSET AS STRING), 7, '0') AS novo_track_id

FROM contagem,

UNNEST(GENERATE_ARRAY(0, total * 2)) AS OFFSET
```

```
),

primeiro_id_disponivel AS (

SELECT novo_track_id

FROM possiveis_ids

WHERE novo_track_id NOT IN (SELECT track_id_str FROM todos_ids)

LIMIT 1

),

dados_corrigidos AS (

SELECT

CASE

WHEN track_id_str = '0:00' OR track_id_str IS NULL OR TRIM(track_id_str) = ''
THEN (SELECT novo_track_id FROM primeiro_id_disponivel)

ELSE track_id_str

END AS track_id_corrigido,

REGEXP_REPLACE(

REGEXP_REPLACE(

LOWER(

IF(track_name IS NULL OR TRIM(track_name) = '', 'Track Name Não Informada',
track_name)

),

r'[^a-z0-9\s]', ''

),

r'\s+', ' '

) AS track_name_corrigido,

REGEXP_REPLACE(

REGEXP_REPLACE(
```

```
LOWER(
IF(artist_name IS NULL OR TRIM(artist_name) = '', 'Artist Não Informado',
artist_name)
),
r'[^a-z0-9\s]', ''
),
r'\s+', ' '
) AS artist_name_cleaned,
artist_count,
released_year,
released_month,
released_day,
in_spotify_playlists,
in_spotify_charts,
SAFE_CAST(streams AS INT64) AS streams_corrigidos,
DATE(CAST(released_year AS INT64), CAST(released_month AS INT64),
CAST(released_day AS INT64)) AS release_date
FROM todos_ids
WHERE
released_year > 1930
AND (track_id_str IS NOT NULL AND TRIM(track_id_str) <> '')
AND SAFE_CAST(streams AS INT64) IS NOT NULL
)
SELECT
track_id_corrigido,
track_name_corrigido,
```

```
artist_name_cleaned,

artist_count,

released_year,

released_month,

released_day,

in_spotify_playlists,

in_spotify_charts,

streams_corrigidos,

release_date

FROM dados_corrigidos

;
```

Dessa forma, realizei a limpeza dos dados, onde foram removidos 8 track_ids por se tratarem de dados duplicados relacionados entre track_name e artist_name.
Como informado anteriormente, os pares desses duplicados apresentavam dados inconsistentes, por isso ambos os pares foram removidos.
A query também tratou os 2 valores nulos encontrados em track_name, substituindo-os por “Track Name Não Informada”.
Foram removidos caracteres especiais, espaços em excesso e padronizados os dados de algumas strings.
O valor fora do padrão em track_id foi transformado em uma sequência numérica única de 7 dígitos.
A linha que continha dados incorretos, com data de lançamento sendo 1930, foi removida, mantendo apenas registros com ano superior a 1930.
A variável streams, que havia sido interpretada como **string**, foi convertida para **INTEGER**, e a linha que continha o dado inválido foi excluída.
Foi criada uma nova variável com a data completa de lançamento da música no.

Após o processo de limpeza, a tabela foi reduzida de 953 para 943 linhas, refletindo a remoção de registros duplicados e inconsistentes.

Para a view: **“view_2track_in_competition_limpa”**, da tabela **“2track_in_competition”** foi utilizado:

```
CREATE OR REPLACE VIEW
projeto2-hipoteses-459613.spotify_2023.view_2track_in_competition_limpa AS
```

```sql
WITH todos_ids AS (

SELECT *,

CAST(track_id AS STRING) AS track_id_str

FROM `projeto2-hipoteses-459613.spotify_2023.2track_in_competition`

),

dados_corrigidos AS (

SELECT

CASE

WHEN track_id_str = '00:00' THEN '1000000'

ELSE track_id_str

END AS track_id_corrigido,

in_apple_playlists,

in_apple_charts,

in_deezer_playlists,

in_deezer_charts

FROM todos_ids

)

SELECT

track_id_corrigido,

in_apple_playlists,

in_apple_charts,

in_deezer_playlists,

in_deezer_charts

FROM dados_corrigidos dc

WHERE track_id_corrigido IN (
```

```
SELECT track_id_corrigido

FROM `projeto2-hipoteses-459613.spotify_2023.view_1track_in_spotify_limpa`

);
```

Foi excluída a coluna “in_shazam_charts”, que continha 50 valores nulos; o track_id 00:00 foi transformado em 1000000, e foi realizada a filtragem de dados, mantendo apenas os registros cujo track_id_corrigido também estava presente na: view_1track_in_spotify_limpa.

Antes da limpeza e importação dos dados no BigQuery, a tabela continha 953 linhas. Após a remoção de 79 registros que apresentavam valores com casas decimais na coluna “in_deezer_playlists”, o total foi reduzido para 874 linhas. Posteriormente, com as etapas de limpeza e filtragem realizadas no BigQuery, essa tabela passou a conter 864 linhas.

Para a view: **“view_3track_technical_info_limpa”**, da tabela **“3track_technical_info”** foi utilizado:

```
CREATE OR REPLACE VIEW
`projeto2-hipoteses-459613.spotify_2023.view_3track_technical_info_limpa` AS

WITH dados_corrigidos AS (

SELECT

CASE

WHEN CAST(track_id AS STRING) = '0:00' THEN '1000000'

ELSE CAST(track_id AS STRING)

END AS track_id_corrigido,

bpm,

mode,

danceability_pct,

valence_pct,

energy_pct,

acousticness_pct,

instrumentalness_pct,
```

```
liveness_pct,

speechiness_pct

FROM `projeto2-hipoteses-459613.spotify_2023.3track_technical_info`

)

SELECT *

FROM dados_corrigidos

WHERE track_id_corrigido IN (

SELECT DISTINCT track_id_corrigido

FROM `projeto2-hipoteses-459613.spotify_2023.view_2track_in_competition_limpa`

);
```

Para essa view, foi excluída a coluna **“key”**, que continha 95 valores nulos.

O track_id 0:00 foi transformado em 1000000 e foi realizado também a filtragem, dessa vez mantendo apenas os registros cujo track_id_corrigido também estava presente na: view_2track_in_competition_limpa.

Como resultado, a view final ficou com 864 linhas.

Embora a query fique salva nos detalhes da view, a consulta utilizada para a criação de cada uma delas também foi salva separadamente, a fim de facilitar futuras alterações, se necessário.

Antes de unir as tabelas, considerei mais adequado iniciar a junção pela “view_1track_in_spotify_limpa”.

Para facilitar a unificação, padronizei a quantidade de linhas, já que essa era a única view que continha mais registros do que as demais.

A **“view_1track_in_spotify_padronizada_limpa”** foi criada utilizando a seguinte query:

```
CREATE OR REPLACE VIEW
`projeto2-hipoteses-459613.spotify_2023.view_1track_in_spotify_padronizada_lim
pa` AS

SELECT a.*

FROM `projeto2-hipoteses-459613.spotify_2023.view_1track_in_spotify_limpa` a
```

```
INNER JOIN
`projeto2-hipoteses-459613.spotify_2023.view_2track_in_competition_limpa` b

ON a.track_id_corrigido = b.track_id_corrigido;
```

Para realizar a unificação das tabelas, aproveitei esta etapa para incluir a variável **“participacao_total_playlists”** referente à meta 5.1.8, que ainda estava pendente, já que imaginei que este seria o melhor momento para criá-la.

Essa nova variável foi calculada pela soma das colunas in_spotify_playlists, in_apple_playlists e in_deezer_playlists, presentes em views diferentes.

A inclusão foi feita diretamente na query de criação da tabela unificada, facilitando a consolidação e o acesso aos dados limpos.

A query utilizada foi a seguinte:

```
CREATE TABLE
`projeto2-hipoteses-459613.spotify_2023.tabela_unificada_tracks_filtrada` AS

SELECT

v1.track_id_corrigido,

v1.* EXCEPT (track_id_corrigido),

v2.* EXCEPT (track_id_corrigido),

v3.* EXCEPT (track_id_corrigido),

--- Variável de participação total nas playlists

IFNULL(v1.in_spotify_playlists, 0)

+ IFNULL(v2.in_apple_playlists, 0)

+ IFNULL(v2.in_deezer_playlists, 0) AS participacao_total_playlists

FROM
`projeto2-hipoteses-459613.spotify_2023.view_1track_in_spotify_padronizada_lim
pa` v1

LEFT JOIN
`projeto2-hipoteses-459613.spotify_2023.view_2track_in_competition_limpa` v2
ON v1.track_id_corrigido = v2.track_id_corrigido
```

```
LEFT JOIN
`projeto2-hipoteses-459613.spotify_2023.view_3track_technical_info_limpa` v3
ON v1.track_id_corrigido = v3.track_id_corrigido;
```

2.Tabela Unificada Expandida

Para criar a **“tabela_unificada_tracks_expandida”**, eu reutilizei a primeira view "view_1track_in_spotify_limpa", e elaborei uma nova view: "view_3track_technical_info_expandida", realizando uma pequena modificação na query. Com essas views prontas, utilizei a seguinte query para gerar a tabela:

```
 CREATE TABLE
`projeto2-hipoteses-459613.spotify_2023.tabela_unificada_tracks_expandida` AS

SELECT

v1.track_id_corrigido,

v1.* EXCEPT (track_id_corrigido),

v2.* EXCEPT (track_id_corrigido),

FROM `projeto2-hipoteses-459613.spotify_2023.view_1track_in_spotify_limpa` v1

LEFT JOIN
`projeto2-hipoteses-459613.spotify_2023.view_3track_technical_info_expandida`
v2 ON v1.track_id_corrigido = v2.track_id_corrigido
```

Esta tabela foi elaborada sem incluir os dados da tabela “2track_in_competition” resultando em um total de **943 linhas**, após a exclusão de **8 registros duplicados** (combinando track_name e artist_name), **1 linha com valor incorreto no campo stream** (registrado como string) e **1 linha cujo ano de lançamento foi considerado inválido**.

Por conter um volume maior de dados, essa tabela será especialmente útil em hipóteses que não exigem o uso de variáveis como in_deezer_playlists, permitindo análises mais robustas do ponto de vista estatístico e com maior representatividade dos dados disponíveis.

Em resumo, conforme mencionado anteriormente, para conduzir a análise com o objetivo de confirmar ou refutar as hipóteses levantadas pela gravadora, farei uso de duas tabelas: *“tabela_unificada_tracks_filtrada”* e *“tabela_unificada_tracks_expandida”*, selecionando a mais adequada de acordo

com a necessidade de variáveis específicas e a quantidade de registros disponíveis para cada hipótese.

5.1.10 🔵

Para calcular o número total de músicas de artistas solo, considerando apenas as músicas individuais de cada artista, utilizei uma tabela temporária e executei a seguinte query:

```
WITH artistas_solo AS (

SELECT

artist_name_cleaned,

track_name_corrigido

FROM `spotify_2023.tabela_unificada_tracks_expandida`

WHERE artist_count = 1

),

total_musicas_solo_por_artista AS (

SELECT

artist_name_cleaned,

COUNT(DISTINCT track_name_corrigido) AS total_musicas_solo

FROM artistas_solo

GROUP BY artist_name_cleaned

)

SELECT

artist_name_cleaned,

total_musicas_solo

FROM total_musicas_solo_por_artista

ORDER BY total_musicas_solo DESC;
```

Como é visível na query acima, foi utilizado a tabela temporária **“tabela_unificada_tracks_expandida”**, pois seu uso fez mais sentido nesse caso.

Essa tabela apresentou 299 artistas solo, com Taylor Swift em primeiro lugar, totalizando 34 músicas solo. Em segundo lugar ficou The Weeknd.

🟪 5.2

Durante a análise exploratória, foram realizadas diferentes abordagens de concatenação de variáveis, com o objetivo de identificar possíveis duplicidades adicionais no conjunto de dados. No entanto, nenhuma duplicidade foi encontrada com base nos critérios testados.

5.2.1 🟣

Utilizando o Power BI, criei duas tabelas matriciais: uma que consolida a quantidade de músicas por artista e outra que mostra a quantidade de músicas por ano de lançamento. Para representar o total geral de músicas, optei por utilizar um cartão (card) em vez de exibir esse número total nas tabelas.

5.2.2 🟣

Foram criados dois gráficos para análise: um gráfico de barras representando a quantidade de músicas por artista, e um gráfico de linhas que apresenta a quantidade e o percentual de músicas lançadas por mês.

A partir da análise mensal, observou-se que janeiro e maio são os meses com maior volume de lançamentos. Das 943 músicas analisadas, 132 foram lançadas em janeiro, representando 14% do total, enquanto 128 foram lançadas em maio, correspondendo a 13,57%. Por outro lado, agosto apresentou o menor número de lançamentos, com apenas 44 músicas, o que equivale a 4,67% do total.

Ao analisar os dados por artista, o cenário apresenta variações. No caso da artista com maior número de músicas na base, Taylor Swift, destaca-se o mês de outubro, responsável por 44,12% de seus lançamentos analisados.

Esses insights revelam que, apesar de haver uma tendência geral de lançamentos mais concentrados no início e no meio do ano, os padrões podem variar significativamente entre os artistas.

5.2.3 🟣

Iniciando a análise pelos streams, utilizei uma tabela matriz, colocando o nome dos artistas nas linhas e, nos valores, a variável referente aos streams em duas colunas:

uma mostrando a média e outra a mediana. Para melhorar a visualização, formatei a variável de streams na tabela principal com separadores de milhares usando pontos.

Ao analisar esses dados, observei que, para a maioria dos artistas, a mediana está abaixo da média, indicando uma distribuição desigual dos valores de streams.

Porém, em alguns casos, como o da banda Blackpink, a média é menor que a mediana, o que pode indicar uma grande variação na popularidade das músicas com mais músicas populares e poucas com menor audiência, que é exatamente o caso dessa banda. Na base analisada, ela possui três músicas: duas mais populares e uma com menor popularidade.

Um exemplo diferente, que é o caso da maioria, é o da artista Taylor Swift, cuja mediana de streams é inferior à média. Isso indica uma distribuição desigual de popularidade entre suas músicas, ao contrário do caso anterior, o sucesso não é consistente em todas as faixas, sendo impulsionado por alguns grandes hits que elevam a média geral.

Para calcular a média e a mediana do total de playlists, realizei duas análises: uma considerando a tabela unificada expandida e outra considerando a tabela unificada filtrada. Para a tabela unificada expandida, na mesma matriz onde eu havia incluído a média e a mediana da variável que representava “streams”, incluí duas vezes a variável que representava o total de playlists do Spotify, uma coluna para a média e outra para a mediana.

Na coluna de média, alguns valores apareceram com casas decimais, já que o cálculo considera a soma dos valores dividida pelo total de músicas do artista, o que pode gerar resultados não inteiros. Mas, a fim de deixar a visualização mais clara, também formatei a coluna que representava as playlists do Spotify na tabela principal de forma que exibisse o ponto como separador de milhar.

Nomeei a primeira página do Power BI como **“Análise Tabela Expandida”** e criei uma nova página com o título **“Análise Tabela Filtrada”**.

Nesta segunda página, inclui um **card** que representa o total de músicas analisadas, com base nos dados da “tabela unificada tracks filtrada". E adicionei uma **tabela matriz** com quatro colunas:

1. Artista
2. Média de Participação Total em Playlists
3. Mediana de Participação Total em Playlists
4. Quantidade total de músicas por artista

A segunda e terceira coluna representa a medida de tendência central (média e mediana) considerando a **soma das três variáveis** relacionadas à presença das músicas em playlists: *in_spotify_playlists, in_apple_playlists* e *in_deezer_playlists.* Também foi realizada a formatação para exibir o ponto como separador de milhar.

Com o intuito de filtrar para realizar uma comparação breve entre as medidas de tendência central nas duas tabelas matriz, inclui a variável que representa a *quantidade total de músicas por artista* na tabela da página **"Análise Tabela Expandida".**

Ao comparar a média e a mediana dos dados referentes apenas às playlists do Spotify com os dados que representam o total de playlists considerando a soma das variáveis: *in_spotify_playlists, in_apple_playlists e in_deezer_playlists*, foi possível observar que os valores seguiram um padrão semelhante entre elas, mantendo a mesma tendência quanto a média e mediana serem mais altas ou mais baixas para os artistas analisados.

O artista Bad Bunny, por exemplo, teve uma *mediana mais baixa do que a média* nas duas tabelas, o que indica, no caso deste artista, que há uma concentração maior de músicas com presença mais modesta em playlists, enquanto algumas faixas com alto número de aparições acabam elevando a média. Isso sugere que, tanto nas playlists do Spotify quanto no total combinado de playlists (in_spotify_playlists, in_apple_playlists e in_deezer_playlists), o sucesso do artista não é igualmente distribuído entre todas as suas músicas, algumas se destacam muito mais do que outras.

De modo geral, assim como na variável "streams", ao calcular a média e a mediana das variáveis relacionadas a playlists, na maioria dos casos, a mediana é inferior à média.

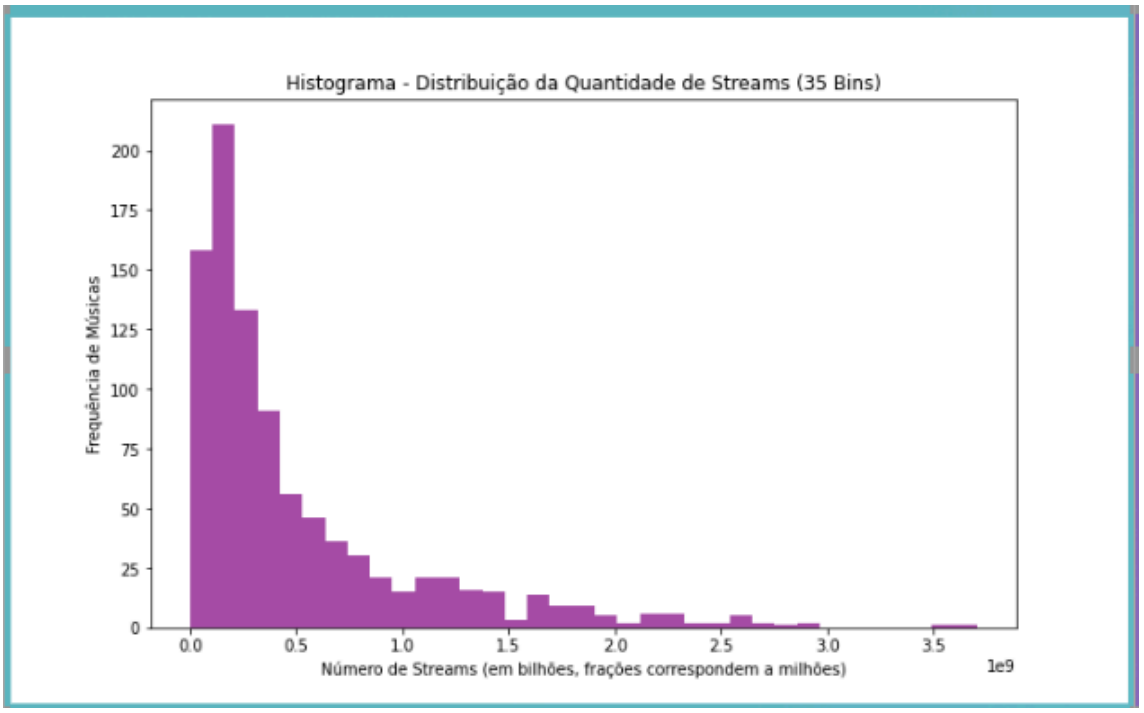
5.2.4 🟣

Criei três histogramas no Power BI com o auxílio de um script em Python.

Para escolher o número ideal de *bins*, utilizei a Regra de Freedman-Diaconis, que leva em conta tanto a variabilidade dos dados quanto o tamanho da amostra. Essa escolha ajudou a representar melhor a distribuição, evitando distorções como excesso ou falta de detalhes no gráfico. Além disso, ajustei o script para que ele exibisse a contagem de bins diretamente no título da visualização, facilitando a interpretação do número de intervalos utilizados.

O **primeiro histograma** foi criado a partir da variável que *representa a quantidade total de streams de cada música*. Para isso, utilizei o seguinte script em Python:

```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

# Dados do Power BI

data = dataset['streams_corrigidos'].dropna()

# Cálculo da Regra de Freedman-Diaconis

q75, q25 = np.percentile(data, [75, 25])

iqr = q75 - q25

bin_width = 2 * iqr / (len(data) ** (1/3))

# Evitar divisão por zero ou bin_width 0

if bin_width == 0: bin_width = 1

# Calcular o número de bins com o bin_width seguro

bin_count = int(np.ceil((data.max() - data.min()) / bin_width))

# Histograma

plt.hist(data, bins=bin_count, color='purple', alpha=0.7)

plt.xlabel('Número de Streams (em bilhões, frações correspondem a
milhões)')

plt.ylabel('Frequência de Músicas')

plt.title(f'Histograma - Distribuição da Quantidade de Streams
({bin_count} Bins)')

# Mostrar o histograma

plt.show()
```

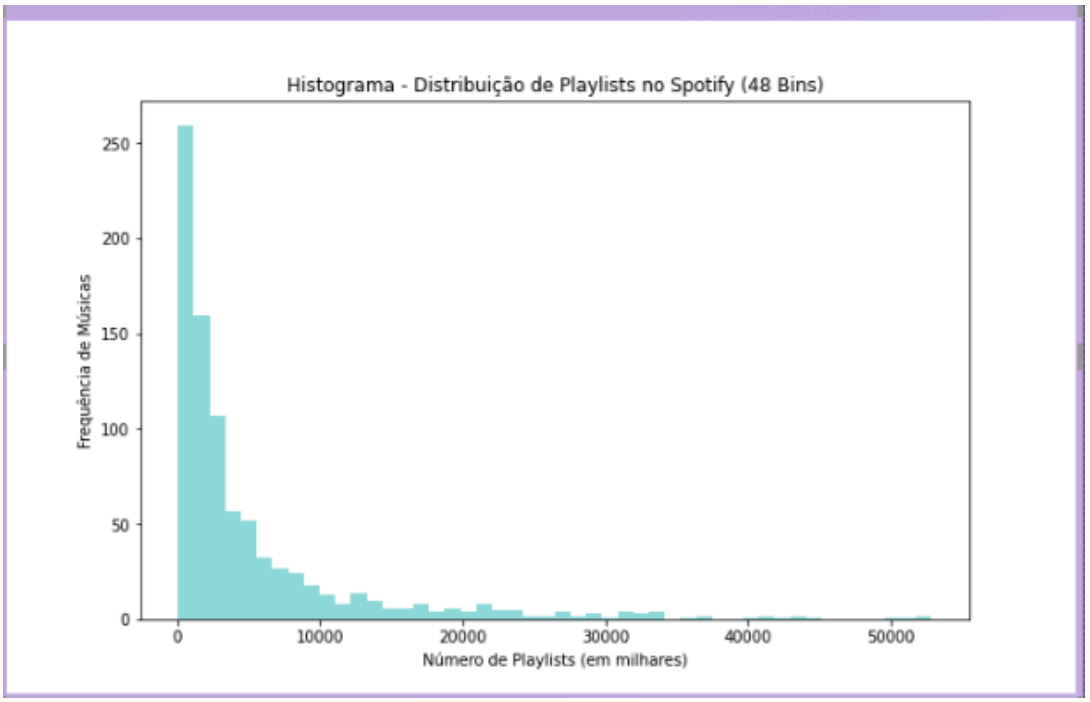


Este histograma mostra uma assimetria acentuada à direita. A maior parte das músicas possui um volume relativamente baixo de reproduções, concentrando-se majoritariamente na faixa entre 0 e 500 milhões de streams. Esse comportamento indica que, dentro do conjunto analisado, poucas músicas alcançam números expressivos de audições, enquanto a maioria permanece com uma audiência moderada ou pequena. Apenas uma pequena parte das músicas atinge cifras bilionárias de streams, ultrapassando, em poucos casos, a marca de 1 bilhão, o que poderíamos chamar de “hits virais”, quando uma música se populariza massivamente e rompe a barreira do consumo comum, alcançando um público global.

O **segundo histograma** foi elaborado a partir da variável que *representa o número de listas de reprodução do Spotify em que cada música está incluída*. Para isso, utilizei o seguinte script em Python:

```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

# Dados do Power BI

data = dataset['in_spotify_playlists'].dropna()

# Cálculo da Regra de Freedman-Diaconis

q75, q25 = np.percentile(data, [75, 25])
```

```
iqr = q75 - q25

bin_width = 2 * iqr / (len(data) ** (1/3))

# Evitar divisão por zero ou bin_width 0

if bin_width == 0: bin_width = 1

# Calcular o número de bins com o bin_width seguro

bin_count = int(np.ceil((data.max() - data.min()) / bin_width))

# Histograma

plt.hist(data, bins=bin_count, color='#5ecbc8', alpha=0.7)

plt.xlabel('Número de Playlists (em milhares)')

plt.ylabel('Frequência de Músicas')

plt.title(f'Histograma - Distribuição de Playlists no Spotify
({bin_count} Bins)')

# Mostrar o histograma

plt.show()
```

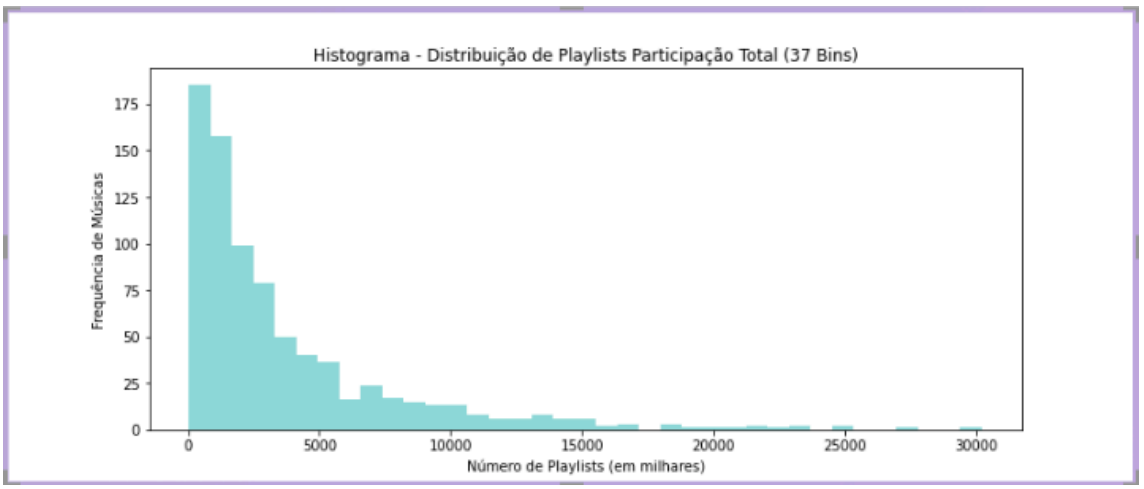


O histograma acima revela uma clara assimetria à direita. A maior parte das músicas está presente em um número relativamente pequeno de playlists do spotify, concentrando-se principalmente abaixo da marca de 10 mil listas. Por outro lado, existe uma longa cauda que se estende até mais de 50 mil playlists, demonstrando que um grupo reduzido de músicas consegue uma inserção massiva nas listas de reprodução.

O **terceiro histograma** foi elaborado na página “Análise Tabela Filtrada” do Power BI, utilizando a variável que *representa o número total de listas de reprodução em que cada música está incluída*, desta vez *considerando a soma das playlists nas plataformas Spotify, Apple Music e Deezer*. Para isso, utilizei o seguinte script em Python:

```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

# Dados do Power BI

data = dataset['participacao_total_playlists'].dropna()

# Cálculo da Regra de Freedman-Diaconis

q75, q25 = np.percentile(data, [75, 25])

iqr = q75 - q25

bin_width = 2 * iqr / (len(data) ** (1/3))

# Evitar divisão por zero ou bin_width 0

if bin_width == 0: bin_width = 1

# Calcular o número de bins com o bin_width seguro

bin_count = int(np.ceil((data.max() - data.min()) / bin_width))

# Histograma

plt.hist(data, bins=bin_count, color='#5ecbc8', alpha=0.7)

plt.xlabel('Número de Playlists (em milhares)')

plt.ylabel('Frequência de Músicas')
```

```
plt.title(f'Histograma - Distribuição de Playlists Participação
Total ({bin_count} Bins)')

# Mostrar o histograma

plt.show()
```



Assim como no histograma anterior, a distribuição geral apresenta novamente uma marcada assimetria à direita. Neste, a maior parte das músicas está inserida em até 5 mil playlists, o que evidencia que, para a maioria, a visibilidade ocorre de forma moderada e segmentada. A semelhança na forma e na tendência entre **os dois histogramas** ambos referentes à distribuição em playlists reforça a consistência desse padrão, mesmo quando os dados são ampliados para incluir múltiplas plataformas, como representado neste terceiro histograma.

Fazendo um **comparativo** do histograma de *streams* com o de *playlists no spotify*, ao analisar artistas de forma individual, a distribuição para alguns foi diferente, como é o exemplo do artista *Bruno Mars*, cujos histogramas mostram uma diferença marcante entre as variáveis, nas músicas solo desse artista. Enquanto os *streams* apresentam uma distribuição assimétrica à direita, com algumas faixas atingindo números extremamente altos de reprodução, os dados que representam as *playlists no spotify* estão concentrados à esquerda, indicando que a maioria das músicas aparece em poucas playlists. Esse contraste sugere que o sucesso em número de streams não depende exclusivamente da inclusão em playlists, e que o artista alcança altos volumes de reprodução mesmo com presença limitada em listas curadas. Isso pode estar relacionado à força da marca pessoal do artista e às buscas diretas dos ouvintes.

5.2.5 🟣

Para aplicar as medidas de dispersão, foi calculado o desvio padrão e a variância para três variáveis: a que representava streams, playlists no spotify e a que representava o total de playlists nas três plataformas (Spotify, Deezer e Apple Music).

Na variável streams, analisando o total geral, este foi o resultado:

Streams			
Média	Mediana	Desvio Padrão	Variância
514.649.334,82	287.278.853	568.702.089,25	323.422.066.314.614.000,00

Valores que indicam uma dispersão extremamente alta, com forte influência de músicas com desempenho muito acima da média.

Referente a playlists no spotify, o resultado foi:

Playlists no Spotify			
Média	Mediana	Desvio Padrão	Variância
5.223,71	2.209	7.927,29	62.841.881,65

O desvio padrão é de 7.927, com variância de aproximadamente 62,8 milhões. A média (5.224) e a mediana (2.209) distantes reforçam a assimetria: poucas músicas aparecem em muitas playlists, enquanto a maioria tem pouca visibilidade.

Para o resultado da Participação Total em Playlists:

Participação Total em Playlists			
Média	Mediana	Desvio Padrão	Variância
3.540,31	1.977	4.244,30	18.014.043,69

A dispersão continua presente, porém menos extrema do que nas variáveis anteriores. É importante destacar que esse total de playlists refere-se à tabela **“tabela_unificada_tracks_filtrada”**, que possui menos linhas analisadas. Essa redução no número de músicas pode ter eliminado os casos mais extremos, tornando o grupo mais homogêneo e, consequentemente, reduzindo a dispersão na participação total em playlists. Além disso, ao considerar várias plataformas, as diferenças tendem a se equilibrar, suavizando as variações extremas observadas quando analisamos apenas o Spotify.

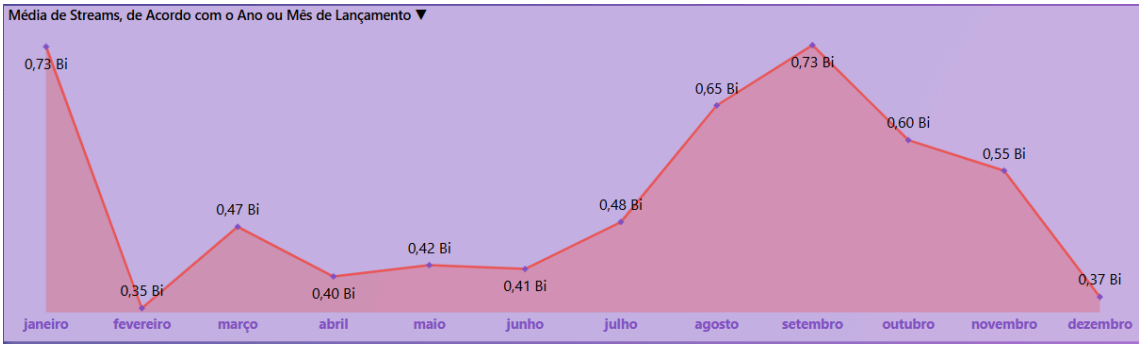
De modo geral o cálculo do desvio padrão e da variância permitiu identificar uma forte concentração de sucesso em poucas faixas, tanto em número de reproduções quanto na presença em playlists, com grande desigualdade entre as músicas analisadas.

5.2.6 🟣

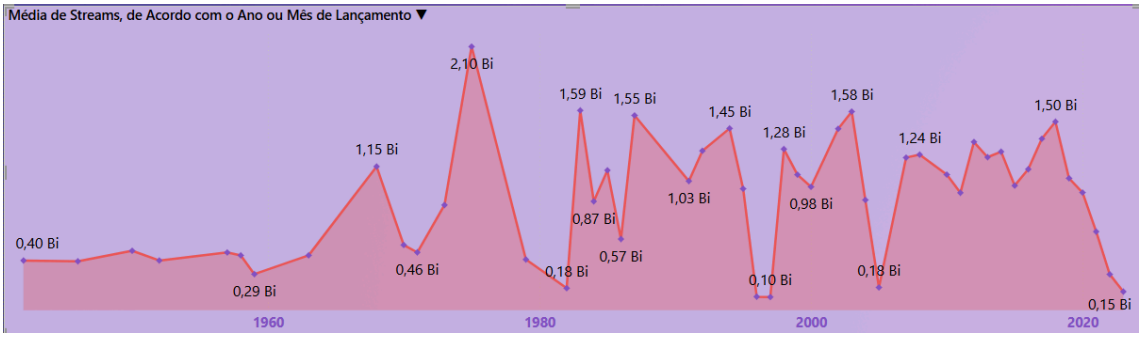
Logo no início, já havíamos criado um gráfico de linhas que representava a quantidade e o percentual de músicas lançadas por mês, considerando os dados da tabela “tabela_unificada_tracks_expandida”. Exibido da seguinte forma:

Aproveitamos esta etapa para criar um gráfico que apresenta a quantidade de streams, de acordo com o *ano ou mês de lançamento*, *conforme o filtro selecionado*. Esse gráfico foi inserido tanto na página “Análise Tabela Expandida” quanto na página “Análise Tabela Filtrada” respeitando os dados específicos de cada uma.

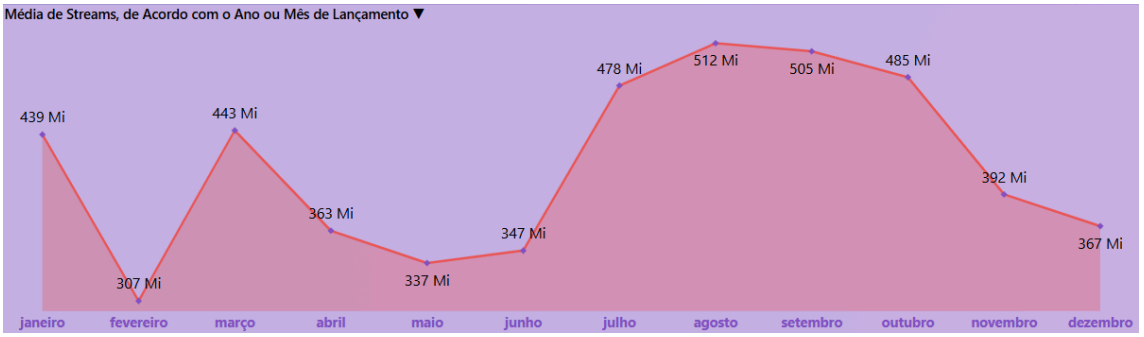
Considerando todas as músicas do conjunto analisado, na página **“Análise Tabela Expandida”** o gráfico foi exibido da seguinte forma (filtro: mês):



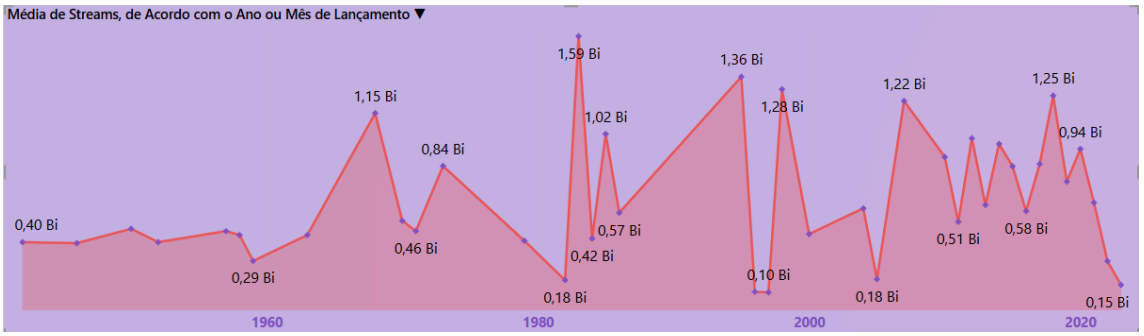
(Filtro: ano):



E em “Análise Tabela Filtrada” (filtro: mês):



(Filtro: ano):



Também replicamos o gráfico de “Quantidade e Percentual de Músicas Lançadas por Mês”, na pagina “Análise Tabela Filtrada”, utilizando os dados da “tabela_unificada_tracks_filtrada”, com o objetivo de possibilitar uma análise comparativa mais precisa.

5.2.7 🟣

Foram calculados os **quartis** das variáveis referentes a *bpm, danceability, valence, energy, acousticness, instrumentality, liveness e speechiness* utilizando SQL no BigQuery. Com base nesses quartis, cada variável foi *inicialmente* classificada em *quatro categorias: Baixa, Média-baixa, Média-alta e Alta.* No entanto, após análise exploratória dos dados, optamos por simplificar a categorização para *três faixas:* ***Baixa, Média e Alta.*** Esse processo foi aplicado em ambas as tabelas: "tabela_unificada_tracks_expandida" e "tabela_unificada_tracks_filtrada".

Após calcular os quartis classificando os dados, as tabelas unificadas de origem foram atualizadas incluindo as novas variáveis. Em seguida, realizamos a atualização no **Power BI** para seguir com as análises.

Código SQL utilizado no BigQuery:

```
CREATE OR REPLACE TABLE `spotify_2023.tabela_unificada_tracks_expandida` AS
```

```sql
SELECT *,

NTILE(4) OVER (ORDER BY bpm) AS bpm_quartil,

CASE NTILE(4) OVER (ORDER BY bpm)

WHEN 1 THEN 'Baixa'

WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS bpm_categoria,

NTILE(4) OVER (ORDER BY energy_pct) AS energy_quartil,

CASE NTILE(4) OVER (ORDER BY energy_pct)

WHEN 1 THEN 'Baixa'

WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS energy_categoria,

NTILE(4) OVER (ORDER BY danceability_pct) AS danceability_quartil,

CASE NTILE(4) OVER (ORDER BY danceability_pct)

WHEN 1 THEN 'Baixa'

WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS danceability_categoria,

NTILE(4) OVER (ORDER BY valence_pct) AS valence_quartil,

CASE NTILE(4) OVER (ORDER BY valence_pct)

WHEN 1 THEN 'Baixa'
```

```
WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS valence_categoria,

NTILE(4) OVER (ORDER BY acousticness_pct) AS acousticness_quartil,

CASE NTILE(4) OVER (ORDER BY acousticness_pct)

WHEN 1 THEN 'Baixa'

WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS acousticness_categoria,

NTILE(4) OVER (ORDER BY instrumentalness_pct) AS instrumentalness_quartil,

CASE NTILE(4) OVER (ORDER BY instrumentalness_pct)

WHEN 1 THEN 'Baixa'

WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS instrumentalness_categoria,

NTILE(4) OVER (ORDER BY liveness_pct) AS liveness_quartil,

CASE NTILE(4) OVER (ORDER BY liveness_pct)

WHEN 1 THEN 'Baixa'

WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS liveness_categoria,
```

```
NTILE(4) OVER (ORDER BY speechiness_pct) AS speechiness_quartil,

CASE NTILE(4) OVER (ORDER BY speechiness_pct)

WHEN 1 THEN 'Baixa'

WHEN 2 THEN 'Média'

WHEN 3 THEN 'Média'

ELSE 'Alta'

END AS speechiness_categoria

FROM `spotify_2023.tabela_unificada_tracks_expandida`;
```

Para a tabela tabela_unificada_tracks_filtrada, foi utilizado o mesmo código, com a única alteração no nome da tabela, substituindo por: `spotify_2023.tabela_unificada_tracks_filtrada`.

5.2.8 🟣

Calculamos o coeficiente de correlação de Pearson entre o número de streams e a quantidade de playlists no Spotify, utilizando o comando CORR no BigQuery. O resultado foi 0,79, indicando uma correlação positiva forte. Isso sugere que músicas adicionadas a mais playlists tendem a receber mais streams. Da mesma forma, músicas com maior número de streams têm mais chances de serem incluídas em playlists, criando um ciclo de reforço mútuo.

Também analisamos a correlação entre o número de streams e a participação total em playlists nas plataformas Spotify, Deezer e Apple Music. O coeficiente foi de 0,80, reforçando a existência de uma associação positiva forte. A presença em playlists nessas plataformas está fortemente ligada a um maior número de reproduções.

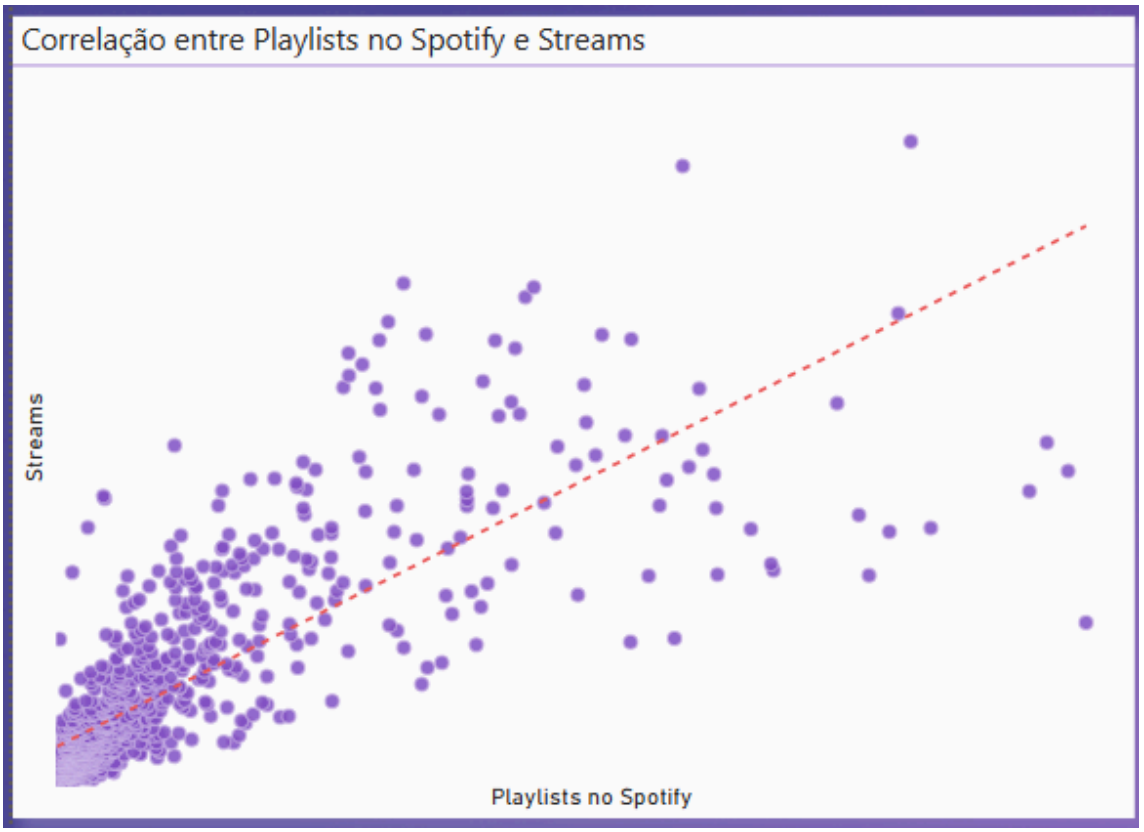
Além disso, calculamos o coeficiente de correlação de Pearson entre o número de streams e a porcentagem de danceabilidade das músicas. Nos dados da tabela expandida, a correlação foi de -0,10, enquanto na tabela filtrada foi de -0,07. Ambos os valores indicam uma correlação negativa muito fraca, praticamente nula. Isso sugere que não há uma relação linear significativa entre o quão dançante é uma música e a quantidade de streams que ela recebe.

Este foi o código utilizado para os cálculos:

```
--- Correlação entre Streams e Playlists no Spotify (Tabela Expandida)

SELECT
```

```
CORR (streams_corrigidos,in_spotify_playlists) as valor_correlacao

FROM `spotify_2023.tabela_unificada_tracks_expandida`; --- Resultado
0.79039702099481768

---Correlação entre Streams e Danceability (Tabela Expandida)

SELECT

CORR (streams_corrigidos,danceability_pct) as valor_correlacao

FROM `spotify_2023.tabela_unificada_tracks_expandida`; --- Resultado
-0.10577319100302494

--- Correlação entre Streams e Total de Playlists (Tabela Filtrada)

SELECT

CORR (streams_corrigidos, participacao_total_playlists) as valor_correlacao

FROM `spotify_2023.tabela_unificada_tracks_filtrada`; --- Resultado
0.80087885574386508

---Correlação entre Streams e Danceability (Tabela Filtrada)

SELECT

CORR (streams_corrigidos, danceability_pct) as valor_correlacao

FROM `spotify_2023.tabela_unificada_tracks_filtrada` --- Resultado
-0.076757763462248038
```

Aproveitei essa etapa e criei um gráfico de dispersão para visualizar a correlação entre playlists no Spotify e streams:



🟥 5.3

5.3.1 🔴

No Power Query, por segurança, primeiro optei por realizar a duplicação das tabelas originais, renomeando as cópias. Selecionei as variáveis correspondentes às características musicais que haviam sido categorizadas, referentes a (bpm, danceability, valence, energy, acousticness, instrumentalness, liveness e speechiness). Em seguida, anulei a dinamização das colunas, o que gerou uma estrutura com uma coluna contendo os nomes das características e outra com suas respectivas categorias (como “alta”, “média” e “baixa”). Com essa base reorganizada, construí uma tabela matriz no Power BI que apresenta a média de streams por categoria para cada característica musical. Esse processo foi realizado para ambas as páginas, "Análise Tabela Expandida" e "Análise Tabela Filtrada", utilizando os dados correspondentes de cada uma.

Em "Análise Tabela Expandida" ficou da seguinte forma:

Média de Streams por Categoria da Característica Musical

Características da Música	Alta	Baixa	Média
acousticness	534.046.869,96	605.088.955,89	459.771.853,18
bpm	524.045.831,17	536.077.686,11	499.256.818,82
danceability	424.267.443,94	587.643.561,97	523.151.679,62
energy	507.252.929,09	551.861.055,53	499.726.006,97
instrumentalness	491.986.670,22	493.203.266,50	536.655.687,15
liveness	490.754.985,50	574.368.068,51	496.686.519,00
speechiness	412.346.673,90	585.860.970,71	529.978.104,40
valence	468.016.386,11	523.009.228,11	533.687.063,90

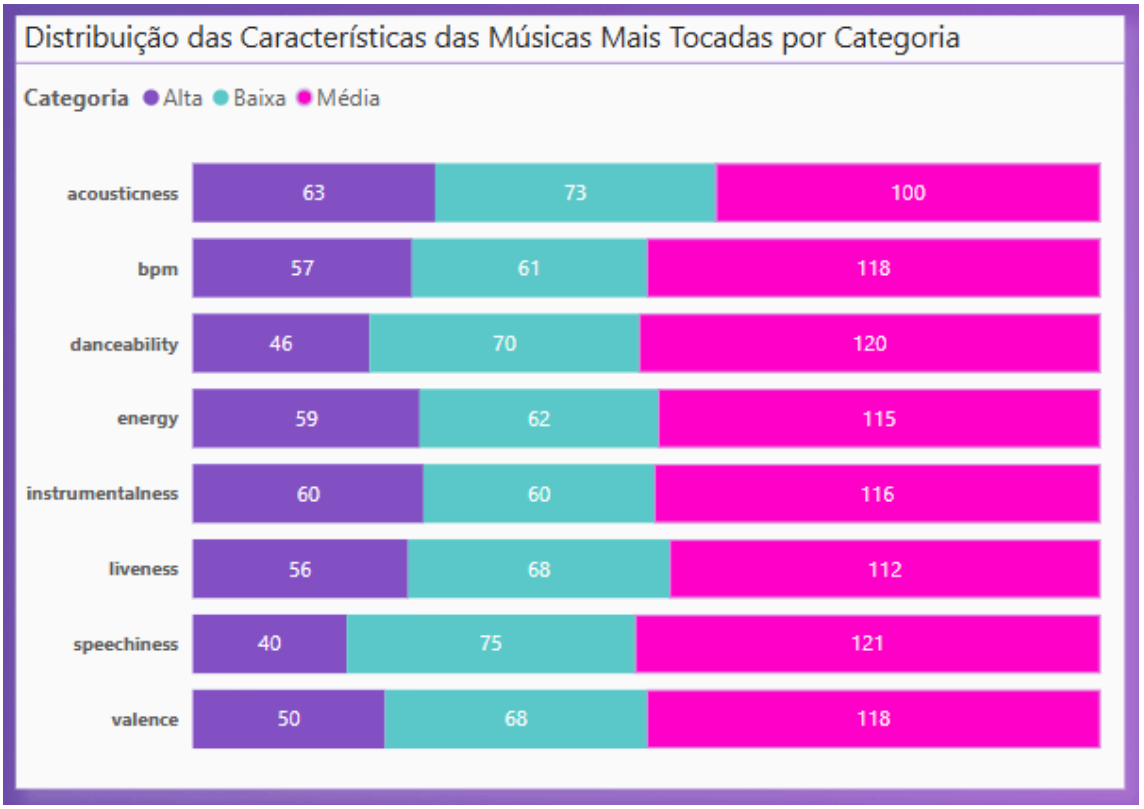
É importante destacar que, ao analisar um artista individualmente, alguns dados podem não aparecer em certas faixas de classificação porque os quartis foram calculados com base no conjunto geral de músicas. Isso significa que, se o artista não tiver músicas suficientes ou que se encaixem nos intervalos definidos pelos quartis, ele pode acabar fora de algumas faixas, mesmo sendo relevante em outros contextos.

Nesta etapa, também aproveitei para criar um gráfico de barras empilhadas com o objetivo de visualizar a distribuição das características das músicas mais tocadas por categoria. Para isso, acessei a guia “Modelagem” e criei uma nova medida utilizando DAX, com a seguinte expressão:

```
Total de Músicas - Top Streams =

VAR Percentil75 =

PERCENTILEX.INC(

ALL(categorias_unpivotadas_tabela_filtrada),

categorias_unpivotadas_tabela_filtrada[streams_corrigidos],

0.75

)

RETURN

CALCULATE(

COUNTROWS(categorias_unpivotadas_tabela_filtrada),

categorias_unpivotadas_tabela_filtrada[streams_corrigidos] >= Percentil75)
```

Em resumo, a medida identificou a quantidade de músicas com maior número de streams ao calcular o valor do percentil 75 (*terceiro quartil,* ***Q3***) da variável “streams”. Em seguida, contou as músicas cujo número de streams é maior ou igual a esse valor, ou seja, aquelas que estão entre os 25% mais tocadas.

Com a nova medida criada, foi possível gerar o gráfico:



A construção das medidas e dos gráficos foi realizada para ambas as tabelas analisadas.

Hipóteses:

5.3.2 🔴

Hipótese 01

- Músicas com BPM (Batidas Por Minuto) mais altos fazem mais sucesso em termos de número de streams no Spotify.

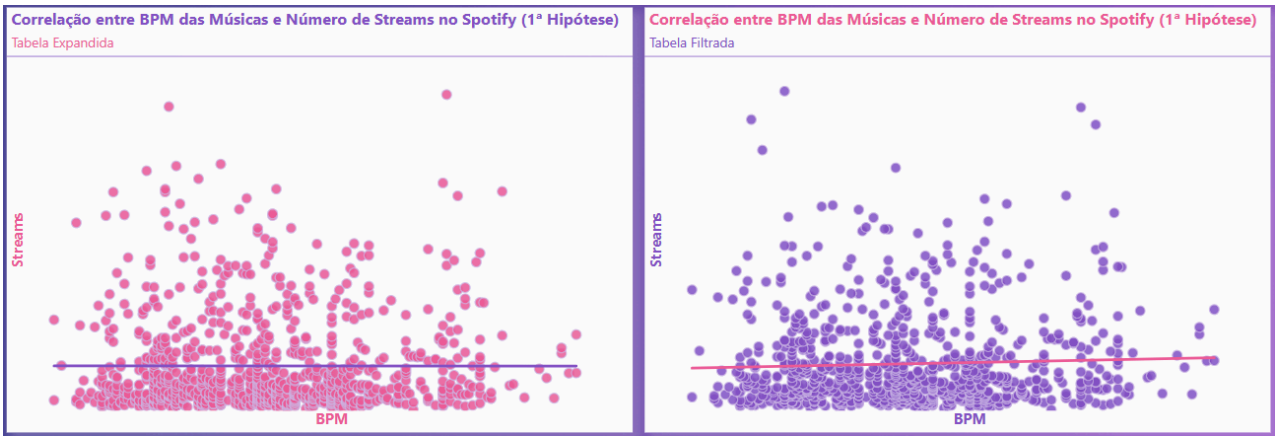
No BigQuery utilizando a consulta:

```
---Hipótese 01 (Com base na tabela expandida)
```

```sql
SELECT

CORR(bpm, streams_corrigidos) AS hipotese_bpm_streams

FROM `spotify_2023.tabela_unificada_tracks_expandida`;
```

Analisamos também com base nos dados da tabela filtrada, substituindo apenas a a tabela para **`spotify_2023.tabela_unificada_tracks_filtrada`**.

Com base na tabela expandida o resultado da correlação foi: **-0.0029889197618340134** enquanto na tabela filtrada o resultado foi: **0.03996845247355893.**



Conclusões Gerais: A correlação entre o BPM (batidas por minuto) das músicas e o número de streams no Spotify apresentou valores próximos de zero em ambas as análises. Apesar de um dos resultados ser positivo, ambos são muito fracos, indicando ausência de relação significativa entre essas variáveis. Portanto, não há evidências consistentes de que músicas com BPM mais alto tenham mais streams.

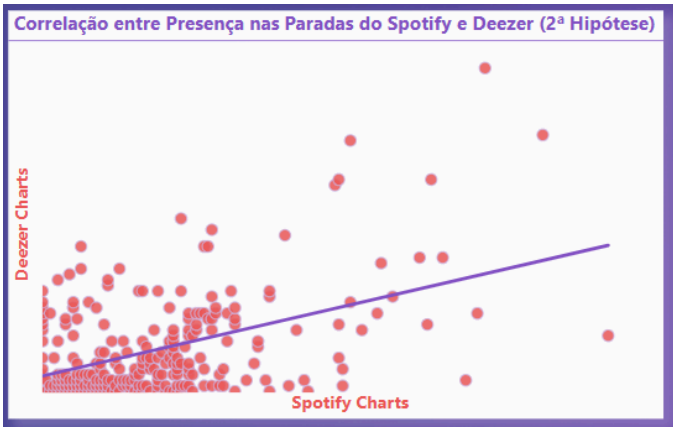
Conclusão da Hipótese: ***Refutada!***

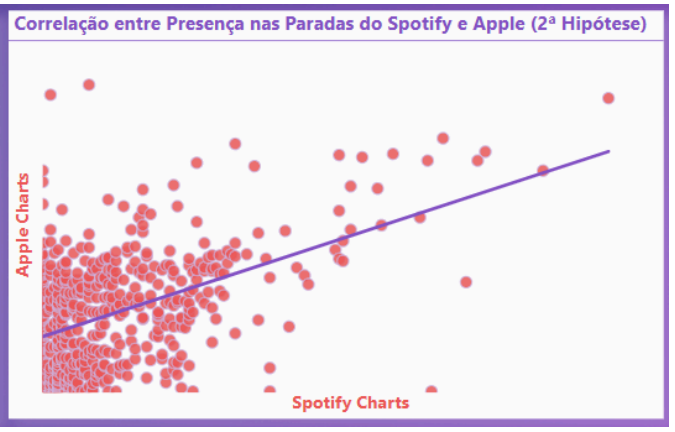
Hipótese 02

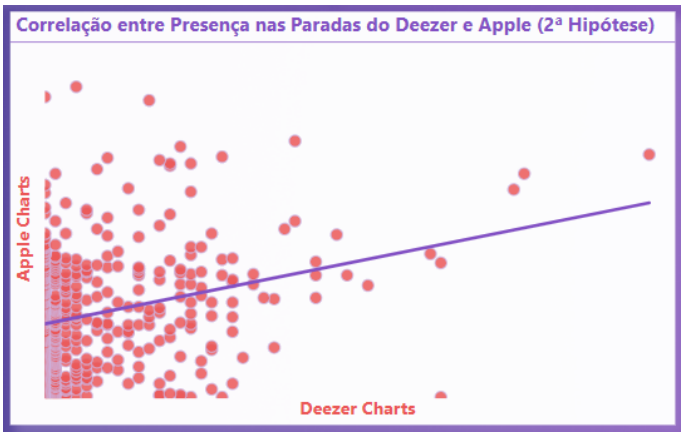
- As músicas mais populares no ranking do Spotify também possuem um comportamento semelhante em outras plataformas, como a Deezer.

Foi utilizada a consulta:

```sql
---Hipótese 02 (Com base na tabela filtrada)

--Deezer

SELECT

CORR (in_spotify_charts, in_deezer_charts) AS hipotese_charts_spotify_deezer,

--Apple

CORR (in_spotify_charts, in_apple_charts) AS hipotese_charts_spotify_apple,

--Deezer e Apple

CORR (in_deezer_charts, in_apple_charts) AS charts_deezer_apple,

FROM `spotify_2023.tabela_unificada_tracks_filtrada`
```







Conclusões Gerais: A correlação entre a presença nas paradas do **Spotify e Deezer** foi de **0.62009213659035722**, e entre **Spotify e Apple 0.58166804588403676**, indicando uma relação moderada e **positiva** que **confirma a hipótese** de que músicas populares no Spotify tendem a ser populares também nessas plataformas. A correlação mais baixa entre **Deezer** e **Apple**: **0.392269215303507** foi incluída *para entender melhor a relação entre as plataformas*, mostrando que o comportamento de popularidade entre Deezer e Apple é menos semelhante, mas, ainda sim positiva o que reforça a importância de comparar múltiplas plataformas para uma análise mais completa.

A **confirmação dessa hipótese** reflete um comportamento consistente dos usuários e tendências compartilhadas no mercado de streaming.

Conclusão da Hipótese: ***Confirmada!***

Hipótese 03

- A presença de uma música em um maior número de playlists está correlacionada com um maior número de streams.

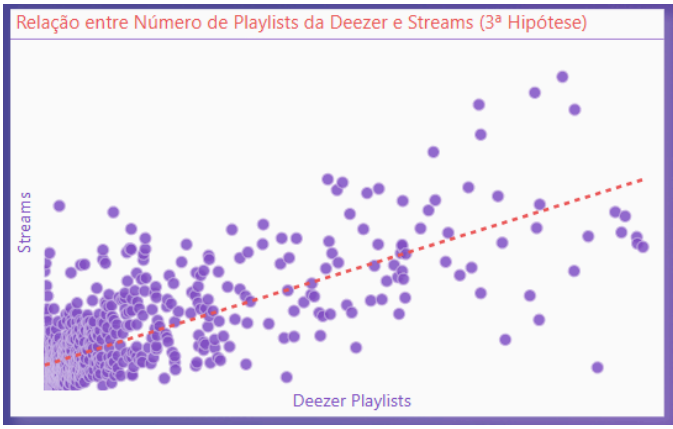
Para essa hipótese, além de **calcular** a correlação **com base no total de playlists**, decidi fazer a análise separada por plataforma, *com o intuito de entender melhor o impacto individual de cada serviço nos streams*, visando realizar uma análise mais detalhada e identificar possíveis diferenças.

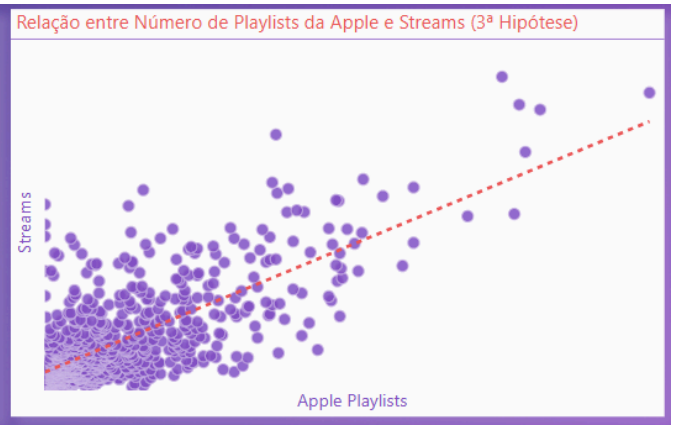
Conforme já consultado anteriormente, com base na tabela expandida o resultado da correlação entre **in_spotify_playlists** e a variável referente a **streams** foi de **0.79039702099481735**. Esse valor indica uma **correlação forte e positiva**, o que **confirma a hipótese** de que músicas presentes em um maior número de playlists tendem a ter mais streams no Spotify. Chegamos a testar também com base nos dados da tabela filtrada e a confirmação da hipótese se manteve, já que o resultado foi de: **0.7943338550056166.**

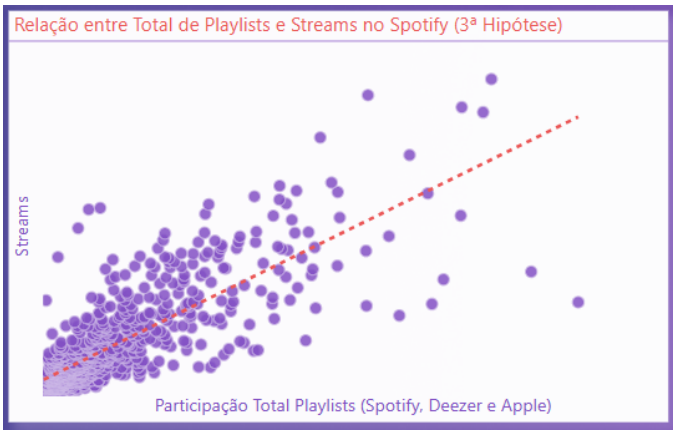
Para as variáveis referentes às playlists da Deezer, da Apple e ao total, considerando a soma das playlists no Spotify, Deezer e Apple, foi realizada a seguinte consulta, com base na tabela filtrada:

```
---Hipótese 03
```

```sql
--Tabela Filtrada para as Variáveis Deezer e Apple

SELECT

CORR(in_deezer_playlists, streams_corrigidos) AS
hipotese_deezer_playlists_streams,

CORR(in_apple_playlists, streams_corrigidos) AS
hipotese_apple_playlists_streams,

CORR(participacao_total_playlists,streams_corrigidos) AS
hipotese_total_playlists_streams

FROM `spotify_2023.tabela_unificada_tracks_filtrada`
```







Conclusões Gerais: O resultado para **Deezer** foi **0.74744255166749163** e para **Apple 0.7235803078709736** . Considerando o número t**otal de playlists**, o resultado foi **ainda maior: 0.80087885574386541** . Esses valores indicam uma relação forte e

positiva entre a presença nas playlists e o número de streams, **confirmando a hipótese** de que *quanto mais playlists uma música aparece, maior é seu sucesso em termos de streams*. Tanto a análise individual por plataforma quanto a do total de playlists mostraram correlações fortes e positivas com os streams, confirmando que o impacto isolado e a soma das exposições multiplataforma influenciam o sucesso da música. Portanto, ambos os aspectos confirmam a hipótese.

Conclusão da Hipótese: ***Confirmada!***

Hipótese 04

- Artistas com um maior número de músicas no Spotify têm mais streams.

Para a quarta hipótese, foram realizadas duas consultas: uma com base na tabela expandida e outra com base na tabela filtrada. A consulta foi estruturada da seguinte forma:

```
---Hipótese 04

--Tabela Expandida

WITH artist_stream_counts_expandida AS (

SELECT

artist_name_cleaned,

COUNT(track_id_corrigido) AS total_songs,

SUM(streams_corrigidos) AS total_streams

FROM

`spotify_2023.tabela_unificada_tracks_expandida`

GROUP BY

artist_name_cleaned

)

SELECT

CORR(total_songs, total_streams) AS hipotese_artists_streams_expandida
```

```
FROM

artist_stream_counts_expandida;


--Tabela Filtrada

WITH artist_stream_counts_filtrada AS (

SELECT

artist_name_cleaned,

COUNT(track_id_corrigido) AS total_songs,

SUM(streams_corrigidos) AS total_streams

FROM

`spotify_2023.tabela_unificada_tracks_filtrada`

GROUP BY

artist_name_cleaned

)

SELECT

CORR(total_songs, total_streams) AS hipotese_artists_streams_filtrada

FROM

artist_stream_counts_filtrada;
```

**Para gerar o gráfico de dispersão desta hipótese, foram criadas duas novas tabelas no BigQuery: uma baseada na tabela expandida e outra na tabela filtrada.*

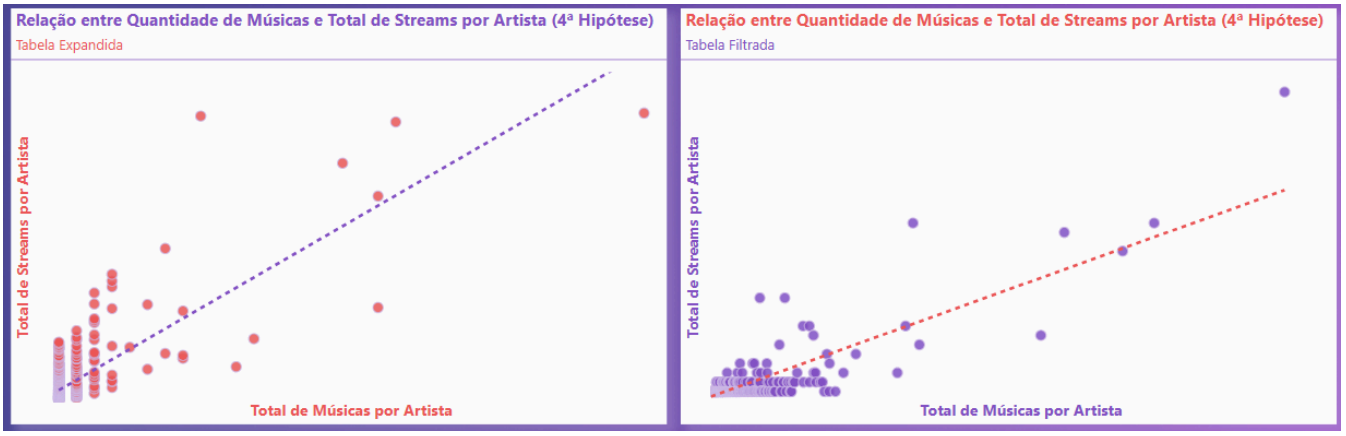
A consulta utilizada para a tabela expandida foi:

```
CREATE TABLE
`projeto2-hipoteses-459613.spotify_2023.total_streams_total_songs_expandida`
AS

SELECT

artist_name_cleaned,
```

```
COUNT(track_id_corrigido) AS total_songs,

SUM(streams_corrigidos) AS total_streams

FROM

`projeto2-hipoteses-459613.spotify_2023.tabela_unificada_tracks_expandida`

GROUP BY

artist_name_cleaned;
```

Para a tabela filtrada, apenas modifiquei os nomes de acordo com a tabela relacionada.

Essa consulta agrupou os dados por artista, calculando o total de músicas e a soma dos streams. Após a criação, essas tabelas foram importadas para o Power BI para a construção dos gráficos.



*Conforme os gráficos acima, embora a linha da tabela expandida seja mais inclinada, os pontos estão mais dispersos, resultando em uma correlação menor. Já na tabela filtrada, mesmo com uma linha menos inclinada, a maior concentração de pontos ao redor dela indica uma correlação mais forte.

Conclusões Gerais: A correlação com base na consulta da **tabela expandida** resultou em **0.776412378316846**, e com base na **tabela filtrada** foi ainda maior: **0.83374340802406421**. Ambos os valores indicam uma correlação forte e positiva entre o número de músicas que um artista possui no Spotify e o total de streams. Esses resultados **confirmam a hipótese** de que artistas com mais músicas tendem a acumular mais streams na plataforma.

Conclusão da Hipótese: ***Confirmada!***

Hipótese 05

- As características da música influenciam o sucesso em termos de número de streams no Spotify.

Para analisar a relação entre as características musicais e o número de streams no Spotify, também foram realizadas duas consultas utilizando a correlação de Pearson.

Com base na **tabela expandida**:

```
---Hipótese 05 (Com base na tabela expandida)

SELECT

CORR(danceability_pct, streams_corrigidos) AS correlacao_danceability,

CORR(valence_pct, streams_corrigidos) AS correlacao_valence,

CORR(energy_pct, streams_corrigidos) AS correlacao_energy,

CORR(acousticness_pct, streams_corrigidos) AS correlacao_acousticness,

CORR(instrumentalness_pct, streams_corrigidos) AS correlacao_instrumentalness,

CORR(liveness_pct, streams_corrigidos) AS correlacao_liveness,

CORR(speechiness_pct, streams_corrigidos) AS correlacao_speechiness

FROM

`spotify_2023.tabela_unificada_tracks_expandida`
```

Que retornou os seguintes resultados:

Correlacao_danceability: -0.10577319100302537.

Correlacao_valence: -0.041889588427309547.

Correlacao_energy: -0.025006456813369951.

Correlacao_acousticness: -0.0051466035601843681.

Correlacao_instrumentalness: -0.043827172699252744.

Correlacao_liveness: -0.05182349974081403.

Correlacao_speechiness: -0.11323445386457354.

Com base na **tabela filtrada**, realizamos a consulta com as mesmas variáveis, substituindo apenas a tabela para: **spotify_2023.tabela_unificada_tracks_filtrada.**

Os resultados foram:

Correlacao_danceability: -0.076757763462248954.

Correlacao_valence: -0.038444234404097145.

Correlacao_energy: -0.04669628794643739.

Correlacao_acousticness: 0.012727584604807678.

Correlacao_instrumentalness: -0.012565633022408965.

Correlacao_liveness: -0.031036011794833691.

Correlacao_speechiness: -0.096694560896720752.

O* **BPM *não foi incluído na análise acima, pois já foi avaliado na hipótese 1, na qual a hipótese foi refutada, porém, foi decidido incluí-lo no conjunto dessas variáveis analisadas com base em suas categorias (baixa, média e alta).*
Para essa análise, utilizamos a linguagem **Python** no **Google Colab***, e a*ntes da aplicação do teste principal, utilizamos o **teste de Shapiro-Wilk** para verificar a normalidade dos dados de **streams** dentro de cada grupo categorizado.
É importante destacar que o Shapiro-Wilk é um teste aplicado **apenas a variáveis numéricas**, como é o caso de “**streams_corrigidos”**. O objetivo foi avaliar se os valores de streams em cada categoria (baixa, média, alta) seguiam uma distribuição normal.

Os resultados do teste Shapiro-Wilk indicaram **distribuições não normais em todos os grupos**, como a análise foi feita com base em 3 categorias, utilizamos o **teste de Kruskal-Wallis** que é o mais adequado para a comparação entre mais de dois grupos independentes e não normais.
O teste **Kruskal-Wallis** foi aplicado para cada uma das características musicais categorizadas **(bpm_categoria, danceability_categoria, valence_categoria, energy_categoria, acousticness_categoria, instrumentalness_categoria, liveness_categoria, speechiness_categoria)**, com o objetivo de verificar se havia diferenças significativas no número de streams entre os grupos definidos como baixa, média e alta.

O Código Python utilizado foi:

```
from scipy import stats

import pandas as pd



# Carregar os dados

df_expandida = pd.read_csv('/content/tabela_expandida.csv')

df_filtrada = pd.read_csv('/content/tabela_filtrada.csv')



variaveis_categoria = [

'bpm_categoria', 'energy_categoria', 'danceability_categoria',
'valence_categoria',

'acousticness_categoria', 'instrumentalness_categoria',
'liveness_categoria', 'speechiness_categoria'

]



def testar_caracteristica(df, nome_tabela):

    print(f"\n====== Testes para {nome_tabela} ======")



    for var_cat in variaveis_categoria:
```

```
        print(f"\nAnalisando variável categorizada: {var_cat}")


# Grupos únicos da categoria (ex: 'baixa', 'média', 'alta')
        grupos = df[var_cat].dropna().unique()


# Valores de streams por grupo
        dados_por_grupo = [df[df[var_cat] ==
g]['streams_corrigidos'].dropna() for g in grupos]


# Testando a normalidade em cada grupo
        normais = []
        for i, grupo in enumerate(grupos):
            stat, p = stats.shapiro(dados_por_grupo[i])
            normais.append(p > 0.05)
            print(f"  Shapiro-Wilk para grupo '{grupo}':
estatística={stat:.4f}, p-valor={p:.4f} -> {'Normal' if p > 0.05
else 'Não normal'}")


# Decidir, baseado na normalidade
        if all(normais):
            # Todos normais -> ANOVA
            stat, p = stats.f_oneway(*dados_por_grupo)
            print(f"  Teste ANOVA: estatística={stat:.4f},
p-valor={p:.4f}")
        else:
# Algum grupo não normal -> Kruskal-Wallis
            stat, p = stats.kruskal(*dados_por_grupo)
```

```
            print(f"  Teste Kruskal-Wallis: estatística={stat:.4f},
p-valor={p:.4f}")



# Interpretar resultado

        if p < 0.05:

            print("  Resultado: Existe diferença significativa no
número de streams entre os grupos dessa característica.")

        else:

            print("  Resultado: Não há diferença significativa no
número de streams entre os grupos dessa característica.")



# Para as duas tabelas

testar_caracteristica(df_expandida, "Tabela Expandida")

testar_caracteristica(df_filtrada, "Tabela Filtrada")
```

Que resultou em:

====== Testes para Tabela Expandida ======

Analisando variável categorizada: bpm_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.7414, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7670, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7634, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=0.8032, p-valor=0.6692

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: energy_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.7474, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7739, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7479, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=0.4730, p-valor=0.7894

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: danceability_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.7885, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7703, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7061, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=5.4094, p-valor=0.0669

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: valence_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.8098, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7182, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7552, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=1.3617, p-valor=0.5062

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: acousticness_categoria

Shapiro-Wilk para grupo 'Alta': estatística=0.7388, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7564, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Baixa': estatística=0.8024, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=8.2370, p-valor=0.0163

Resultado: Existe diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: instrumentalness_categoria

Shapiro-Wilk para grupo 'Alta': estatística=0.7834, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7456, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Baixa': estatística=0.7814, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=0.2394, p-valor=0.8872

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: liveness_categoria

Shapiro-Wilk para grupo 'Alta': estatística=0.7697, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7485, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Baixa': estatística=0.7673, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=5.6266, p-valor=0.0600

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: speechiness_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.8182, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7475, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7397, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=10.2777, p-valor=0.0059

Resultado: Existe diferença significativa no número de streams entre os grupos dessa característica.

====== Testes para Tabela Filtrada ======

Analisando variável categorizada: bpm_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.7528, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7931, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7763, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=1.9587, p-valor=0.3756

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: energy_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.7512, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.8037, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7636, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=3.8887, p-valor=0.1431

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: danceability_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.8058, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7707, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.8030, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=3.8305, p-valor=0.1473

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: valence_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.8312, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7281, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7686, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=0.8886, p-valor=0.6413

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: acousticness_categoria

Shapiro-Wilk para grupo 'Alta': estatística=0.7723, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7698, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Baixa': estatística=0.7956, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=0.7017, p-valor=0.7041

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: instrumentalness_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.7744, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7579, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.8154, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=3.0904, p-valor=0.2133

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: liveness_categoria

Shapiro-Wilk para grupo 'Alta': estatística=0.7585, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7904, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Baixa': estatística=0.7774, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=5.9155, p-valor=0.0519

Resultado: Não há diferença significativa no número de streams entre os grupos dessa característica.

Analisando variável categorizada: speechiness_categoria

Shapiro-Wilk para grupo 'Baixa': estatística=0.8423, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Média': estatística=0.7578, p-valor=0.0000 -> Não normal

Shapiro-Wilk para grupo 'Alta': estatística=0.7391, p-valor=0.0000 -> Não normal

Teste Kruskal-Wallis: estatística=9.4921, p-valor=0.0087

Resultado: Existe diferença significativa no número de streams entre os grupos dessa característica.

Os testes de Kruskal-Wallis revelaram que **apenas duas variáveis: *acousticness* (acústica) e *speechiness* (presença de fala)**, apresentaram **diferenças estatisticamente significativas** no número de streams entre seus grupos **(p < 0,05). Isso sugere que músicas com diferentes níveis dessas características apresentam desempenho distinto em termos de popularidade.**

Como “**Mode**” é uma variável nominal, ela não é adequada para análise por meio da correlação de Pearson. Por isso, foi avaliada utilizando outro teste estatístico. A análise foi realizada em Python, por meio do Google Colab, seguindo os mesmos passos anteriores, inicialmente, aplicamos o teste de Shapiro-Wilk para verificar a normalidade dos dados.
O teste foi realizado tanto na tabela expandida quanto na tabela filtrada, com os dados separados pelos grupos “**Major**” e “**Minor**”. Em todos os casos, o valor de *p* retornado foi significativamente inferior a 0,05, indicando que os dados não seguem uma distribuição normal.
Diante da ausência de normalidade, foi aplicado o **teste de Mann-Whitney U**, que é um **teste não paramétrico** adequado para comparar dois grupos independentes. O objetivo foi verificar se existia diferença significativa no número de streams entre músicas em modo “**Major**” e “**Minor**”.

O Código Python utilizado foi:

```
# Para importar biblioteca necessária

from scipy import stats

import pandas as pd

# Para carregar os arquivos

df_expandida = pd.read_csv('/content/tabela_expandida.csv')

df_filtrada = pd.read_csv('/content/tabela_filtrada.csv')

# APLICANDO O TESTE DE SHAPIRO-WILK - CONSIDERANDO TODOS OS DADOS

shapiro_tudo_expandida =
stats.shapiro(df_expandida['streams_corrigidos'])

shapiro_tudo_filtrada =
stats.shapiro(df_filtrada['streams_corrigidos'])

print("\n SHAPIRO - TODOS OS STREAMS")

print("EXPANDIDA:", shapiro_tudo_expandida)

print("FILTRADA:", shapiro_tudo_filtrada)

# APLICANDO O TESTE DE SHAPIRO-WILK - DIVIDIDO POR MODE

# Separando por grupo (Major / Minor)
```

```
streams_maior_exp = df_expandida[df_expandida['mode'] ==
'Major']['streams_corrigidos']

streams_menor_exp = df_expandida[df_expandida['mode'] ==
'Minor']['streams_corrigidos']

streams_maior_filt = df_filtrada[df_filtrada['mode'] ==
'Major']['streams_corrigidos']

streams_menor_filt = df_filtrada[df_filtrada['mode'] ==
'Minor']['streams_corrigidos']

# Teste de Shapiro-Wilk para cada grupo

shapiro_maior_exp = stats.shapiro(streams_maior_exp)

shapiro_menor_exp = stats.shapiro(streams_menor_exp)

shapiro_maior_filt = stats.shapiro(streams_maior_filt)

shapiro_menor_filt = stats.shapiro(streams_menor_filt)

print("\n SHAPIRO - DIVIDIDO POR MODE")

print("EXPANDIDA - Major:", shapiro_maior_exp)

print("EXPANDIDA - Minor:", shapiro_menor_exp)

print("FILTRADA - Major:", shapiro_maior_filt)

print("FILTRADA - Minor:", shapiro_menor_filt)

from scipy import stats

import pandas as pd

# Ler os dados

tabela_expandida = pd.read_csv('/content/tabela_expandida.csv')

tabela_filtrada = pd.read_csv('/content/tabela_filtrada.csv')

# Separar os streams corrigidos para cada grupo 'mode'

streams_major_exp = tabela_expandida[tabela_expandida['mode'] ==
'Major']['streams_corrigidos']

streams_minor_exp = tabela_expandida[tabela_expandida['mode'] ==
'Minor']['streams_corrigidos']
```

```
streams_major_filt = tabela_filtrada[tabela_filtrada['mode'] ==
'Major']['streams_corrigidos']

streams_minor_filt = tabela_filtrada[tabela_filtrada['mode'] ==
'Minor']['streams_corrigidos']

# Aplicar o teste Mann-Whitney U (não paramétrico)

mannwhitney_exp = stats.mannwhitneyu(streams_major_exp,
streams_minor_exp, alternative='two-sided')

mannwhitney_filt = stats.mannwhitneyu(streams_major_filt,
streams_minor_filt, alternative='two-sided')

# Mostrar resultados

print("Tabela Expandida - Mann-Whitney U Teste:")

print(f"Estatística: {mannwhitney_exp.statistic}, p-valor:
{mannwhitney_exp.pvalue}")

print("\nTabela Filtrada - Mann-Whitney U Teste:")

print(f"Estatística: {mannwhitney_filt.statistic}, p-valor:
{mannwhitney_filt.pvalue}")

# Interpretar resultados

def interpretar_pvalor(p):

    if p < 0.05:

        return "Existe diferença significativa entre os grupos."

    else:

        return "Não há diferença significativa entre os grupos."

print("\nInterpretação Tabela Expandida:",
interpretar_pvalor(mannwhitney_exp.pvalue))

print("Interpretação Tabela Filtrada:",
interpretar_pvalor(mannwhitney_filt.pvalue))
```

Que resultou em:

SHAPIRO - TODOS OS STREAMS

EXPANDIDA: ShapiroResult(statistic=np.float64(0.7607130327224486), pvalue=np.float64(7.277857904328099e-35))

FILTRADA: ShapiroResult(statistic=np.float64(0.7776183907920238), pvalue=np.float64(1.1992466197732194e-32))

SHAPIRO - DIVIDIDO POR MODE

EXPANDIDA - Major: ShapiroResult(statistic=np.float64(0.7712896160108397), pvalue=np.float64(7.593813198124413e-27))

EXPANDIDA - Minor: ShapiroResult(statistic=np.float64(0.7448232801552339), pvalue=np.float64(2.467018249034931e-24))

FILTRADA - Major: ShapiroResult(statistic=np.float64(0.7816678610974156), pvalue=np.float64(3.8629368314851315e-25))

FILTRADA - Minor: ShapiroResult(statistic=np.float64(0.7686649130908313), pvalue=np.float64(1.6627301996561965e-22))

Tabela Expandida - Mann-Whitney U Teste:

Estatística: 114951.5, **p-valor: 0.12009426581104235**

Tabela Filtrada - Mann-Whitney U Teste:

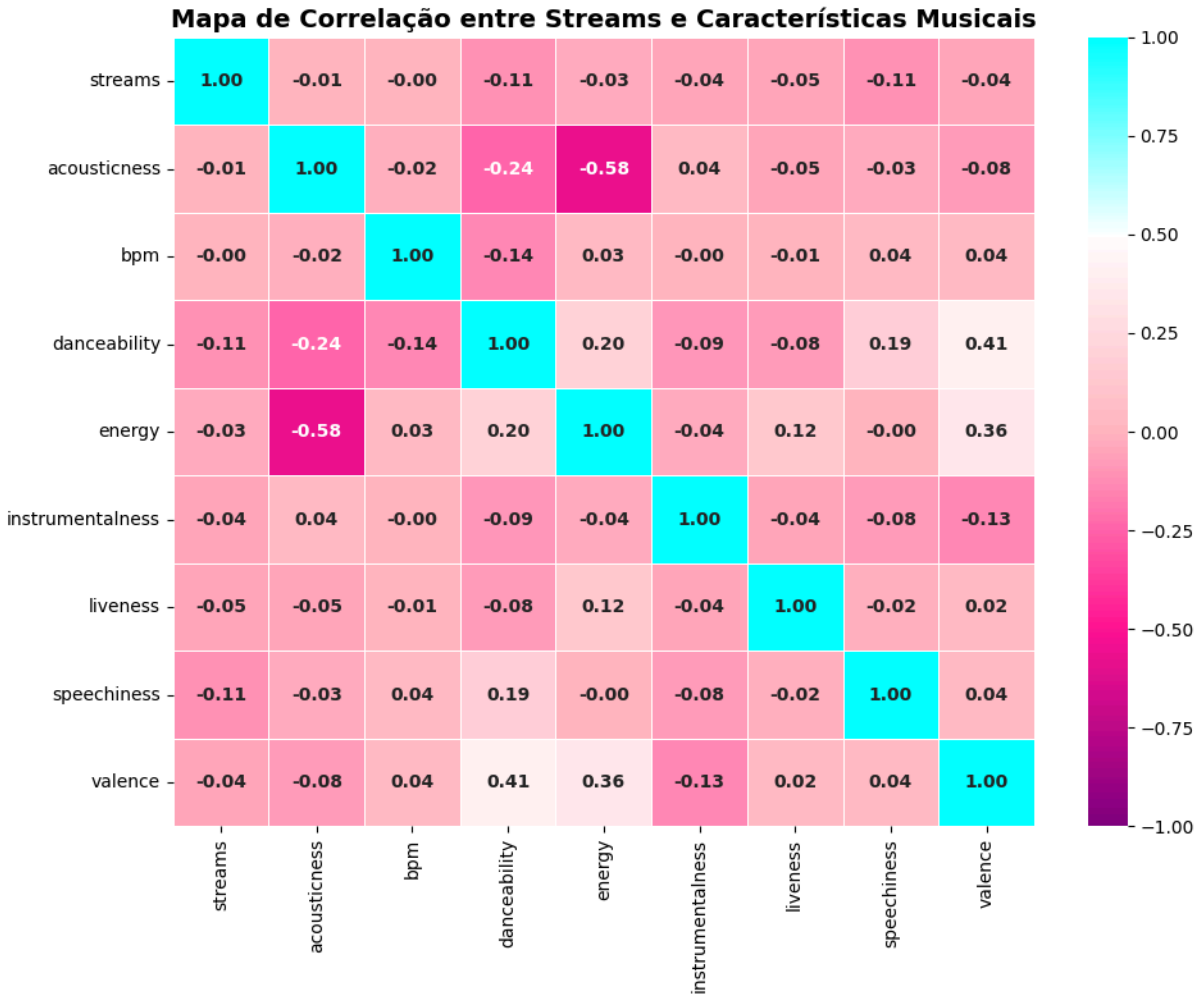
Estatística: 95666.5, **p-valor: 0.2457658412539886**

Interpretação Tabela Expandida: Não há diferença significativa entre os grupos.

Interpretação Tabela Filtrada: Não há diferença significativa entre os grupos.

Como ambos os **p-valores são **superiores a 0,05**, foi concluído que não há diferença estatisticamente significativa entre os grupos. Portanto, com base na variável "mode", não há evidência suficiente para afirmar que o modo musical influencia o número de streams no Spotify.*

O **mapa de calor** abaixo, indica que não há correlações fortes entre as características musicais analisadas e os streams, sugerindo que a popularidade de uma música no Spotify não é diretamente explicada por essas variáveis isoladamente. Contudo, há **correlações moderadas** entre algumas características entre si, como entre energia e positividade, ou acústico e energia, o que pode refletir padrões estilísticos comuns.



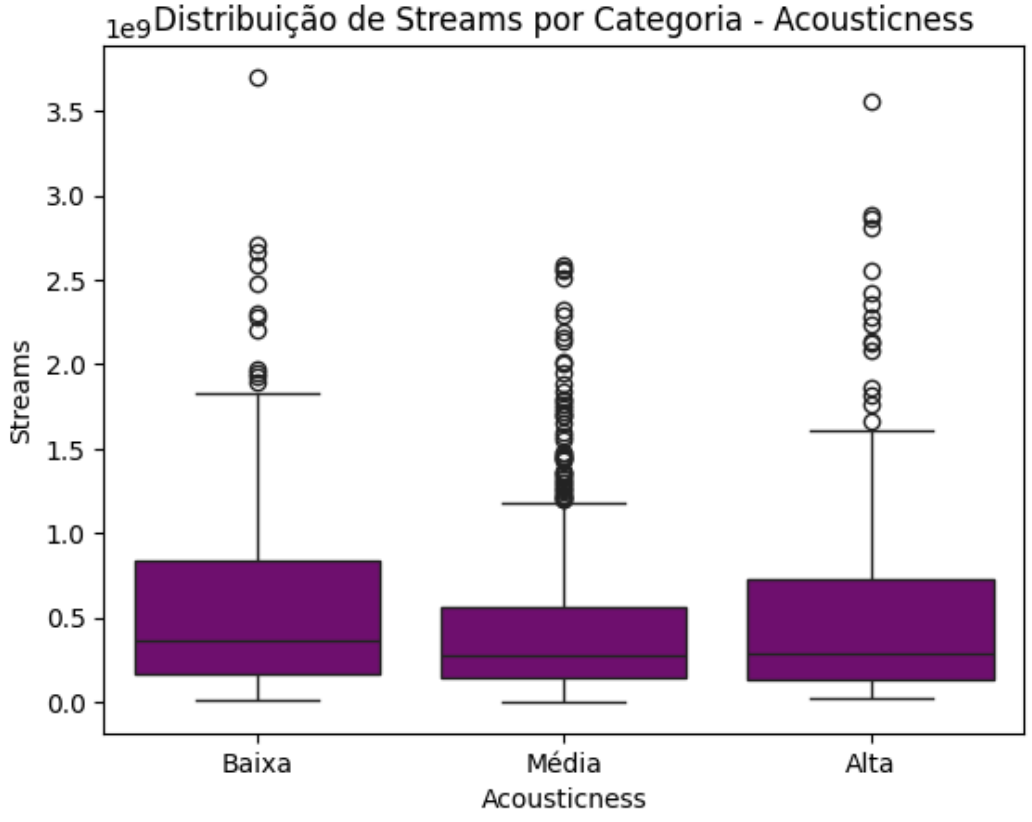
Conclusões Gerais: Na análise baseada na tabela expandida, todas as variáveis apresentaram correlações negativas com os streams. Na tabela filtrada, os resultados foram semelhantes para a maioria das variáveis, com correlações negativas, porém com menor magnitude. A exceção foi a variável **acousticness**, que apresentou correlação negativa muito pequena na tabela expandida (-0.0051466035601843681) e uma correlação positiva, ainda que fraca, na tabela filtrada (0.012727584604807678). Essa divergência na acousticness pode indicar que, ao reduzir o conjunto de dados para a tabela filtrada, aspectos específicos das músicas acústicas ganham uma influência diferente sobre o número de streams, sugerindo variações no perfil das faixas analisadas em cada tabela. Contudo, a diferença encontrada, sendo tão baixa, não é forte o suficiente para indicar uma influência real de acousticness no número de streams.

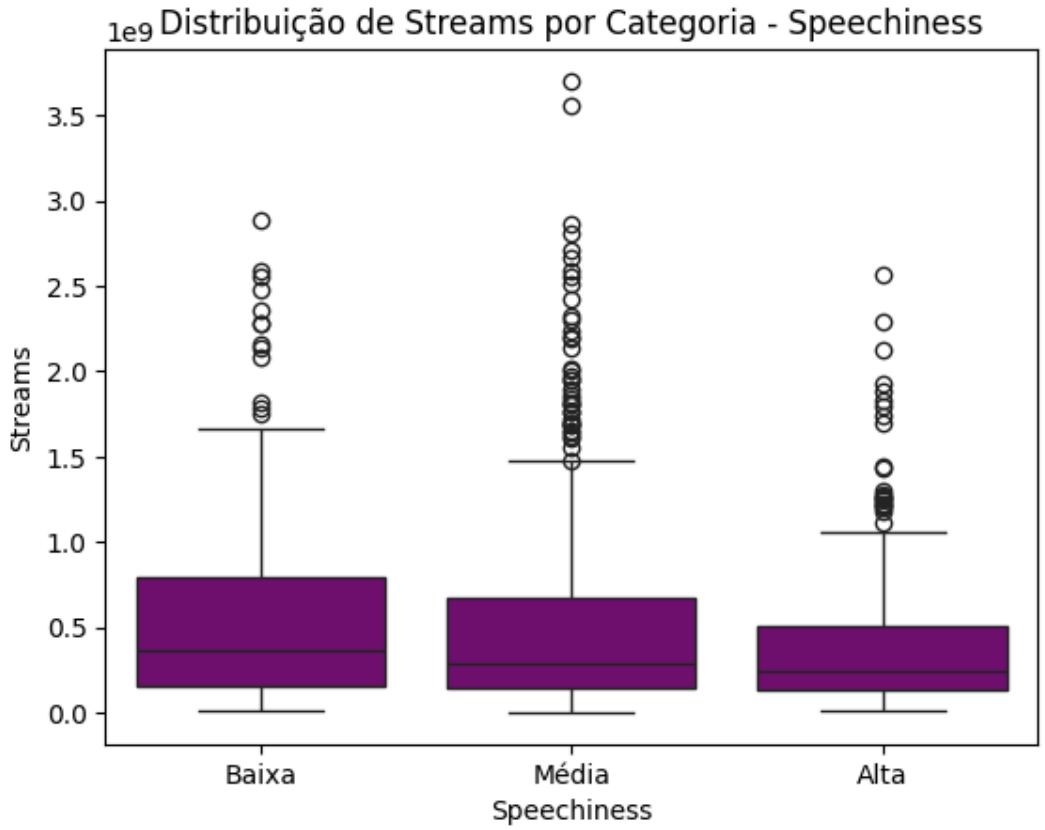
De modo geral, considerando o teste de Pearson, todas as correlações observadas foram muito fracas, negativas ou próximas de zero, indicando que as características musicais não apresentam uma relação linear significativa com o sucesso em número

de streams no Spotify. Com base nesses resultados, a hipótese de que tais características influenciam diretamente o desempenho das músicas **foi refutada.**

No entanto, a *análise complementar* por meio do teste de **Kruskal-Wallis** revelou **evidências relevantes:** embora a maioria das variáveis categorizadas (como *bpm*, *energy* e *valence*) não tenha apresentado diferenças significativas entre os grupos de streams, duas características: ***acousticness*** **e** ***speechiness*** mostraram resultados estatisticamente significativos. Isso sugere que certos aspectos específicos da composição musical podem, de fato, estar associados ao sucesso.

Speechiness_categoria demonstrou uma associação estatisticamente significativa com o número de streams em ambas as bases de dados, sugerindo uma relação mais robusta.
Já a variável **acousticness_categoria** apresentou significância apenas na tabela expandida, o que pode indicar uma relação mais fraca ou dependente de um maior volume de dados para ser evidenciada.
Os gráficos de **boxplot** a seguir apresentam a distribuição do número de streams por categoria (Baixa, Média e Alta) para ambas as variáveis. Observa-se que, em ambos os casos, a categoria **Baixa** está associada a valores mais altos de streams, tanto em média quanto em mediana, reforçando a interpretação de que **músicas com menor speechiness e acousticness tendem a performar melhor** nas plataformas de streaming.





Portanto, embora a hipótese principal tenha sido amplamente refutada com base nas correlações, ela **não pode ser totalmente descartada**. Os achados indicam que o sucesso musical depende de múltiplos fatores, e que características como ***acousticness*** **e principalmente** ***speechiness*** *podem desempenhar um papel relevante, merecendo maior atenção em estudos futuros. Além disso, o* ***mapa de calor*** *revelou que* ***músicas acústicas tendem a ser menos enérgicas****,* ***músicas mais dançantes costumam soar mais positivas*** *e* ***músicas mais energéticas também tendem a ser mais positivas****.*

Conclusão da Hipótese*: Refutada em termos gerais, mas com ressalvas relevantes.*

Marco Adicional:

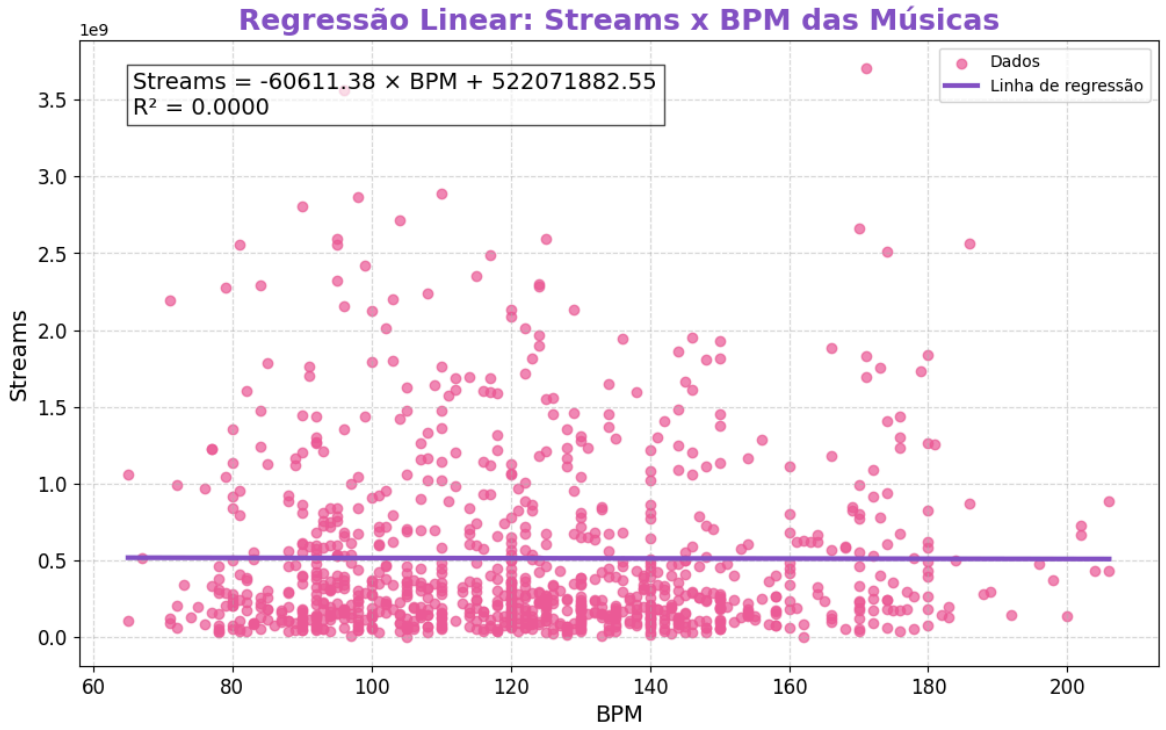
Utilizando Python no Google Colab, foi aplicada uma regressão linear para verificar a **primeira** e a **terceira hipótese**.

Para a **primeira hipótese**, o resultado obtido foi:

```
Equação da regressão: streams_corrigidos = -60611.38 × bpm + 522071882.55
Coeficiente de determinação R²: 0.0000
```

Esse valor de R² indica que não há relação linear significativa entre o BPM das músicas e o número de streams.

Conforme demonstrado no gráfico abaixo:



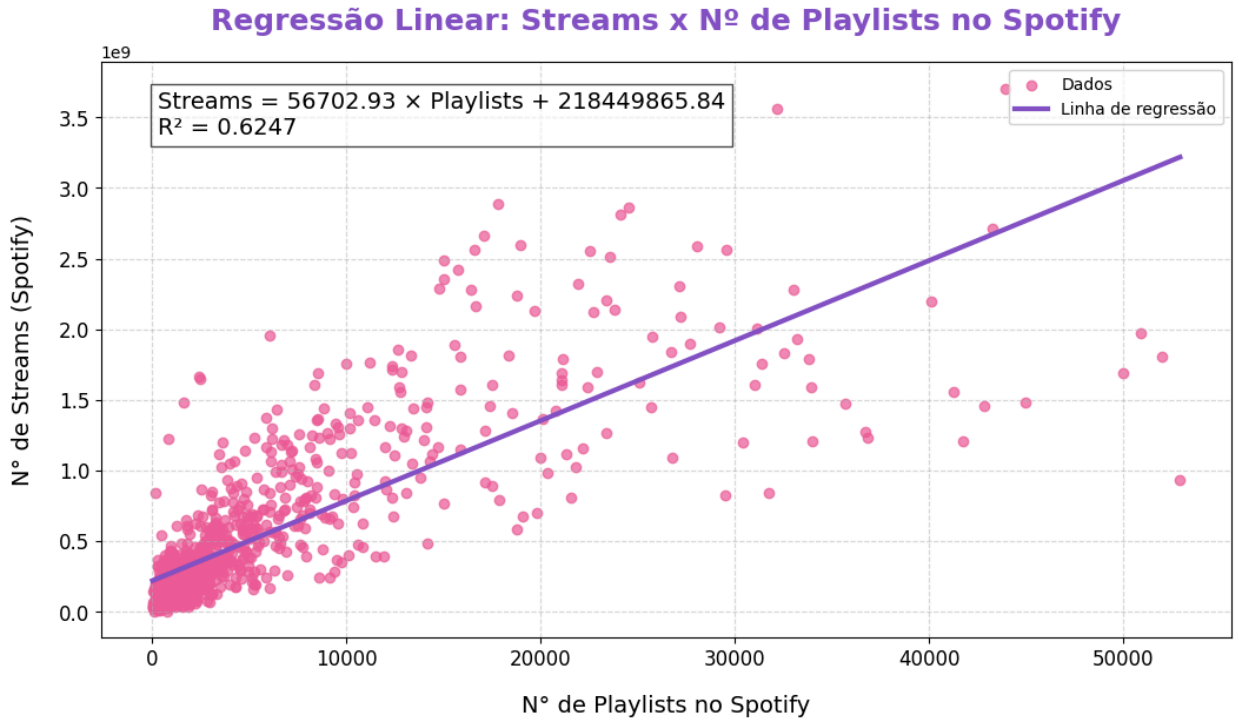
O gráfico acima mostra que a linha de tendência tem uma inclinação negativa, o que indica que quanto maior o BPM, menor tende a ser o número de streams. No entanto, essa relação é muito fraca e os dados estão bastante dispersos.

Não há uma relação forte ou clara entre BPM e número de streams.

Para a **terceira hipótese**, considerando *apenas as playlists do Spotify*, obtivemos a seguinte equação de regressão linear:

```
Equação da regressão: streams_corrigidos = 56702.93 × in_spotify_playlists + 218449865.84
Coeficiente de determinação R²: 0.6247
```

Conforme demonstrado no gráfico abaixo:



O coeficiente de determinação (R^2) foi de 0,6247, indicando que aproximadamente 62,47% da variação total no número de streams pode ser explicada pelo número de playlists em que a música está presente no Spotify.

Esse resultado evidencia uma relação positiva, moderada e significativa entre a presença em playlists e o desempenho em streams, reforçando a importância das playlists como fator de sucesso no consumo musical na plataforma.

Resumo de Resultados e Conclusões:

De modo geral, a análise das cinco hipóteses revelou que o sucesso no Spotify e em outras plataformas de streaming depende de múltiplos fatores, muitos deles não óbvios. Não foi encontrada relação significativa entre o BPM ou características musicais tradicionais e o número de streams, indicando que ritmo e atributos gerais isolados não determinam o sucesso de uma música.

Por outro lado, constatou-se uma forte correlação entre a popularidade consistente em plataformas como Spotify, Deezer e Apple Music, a ampla presença em playlists e o maior volume de músicas lançadas pelo artista, evidenciando a importância da exposição e de uma estratégia multiplataforma bem estruturada.

O mapa de calor apontou que músicas acústicas tendem a ser menos enérgicas, enquanto faixas mais dançantes e energéticas geralmente apresentam tonalidades mais positivas. Esses insights sugerem que músicas com maior energia e positividade têm mais potencial para atrair e engajar o público.

Além disso, características específicas como menor “speechiness” (menos presença de fala) e menor “acousticness” (menos elementos acústicos) estão associadas a um desempenho melhor, reforçando a preferência por músicas mais dinâmicas e menos acústicas.

Diante disso, recomenda-se que a gravadora desenvolva um repertório diversificado e numeroso para o novo artista, priorizando a ampla distribuição em playlists relevantes e a presença consistente em múltiplas plataformas de streaming. Investir em músicas com maior energia, positividade e menor acústica pode aumentar as chances de sucesso. Por fim, uma estratégia integrada, que combine essas características musicais com ações para maximizar a exposição, será fundamental para potencializar o lançamento e consolidar o artista no mercado.

Links de Interesse:

- ***Apresentação de Slides***

https://docs.google.com/presentation/d/1DokC3usoecNHDnz8J-lH2Wrt5xUCVe2kDpm8JmwyUIw/edit?usp=sharing

- ***GitHub***

  Gabriela: *https://github.com/GabrielaGTech*