UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS
PROJETO INTEGRADOR DO MÓDULO 3

INTRODUÇÃO

O presente documento tem o objetivo de descrever o projeto envolvendo as disciplinas estudadas durante o módulo 3 do curso de especialização de Ciência de Dados pela UTFPR.

O projeto tem a proposta transversal de trabalhar todo o conteúdo abordado nas disciplinas de *Introdução ao Big Data*, *Processamento Analítico de Dados*, *Recuperação de Informação Baseada em Conteúdo*.

O trabalho foi desenvolvido pela Equipe 3, composta por:

- Arturo Vaine
- Írio Moesch
- Robson Mamede

CONTEXTO

A proposta desenvolvida aqui objetiva a utilização da base de dados para a criação de um Data Warehouse com informações referentes a músicas fornecidas pela API de desenvolvimento do Spotify.

Os dados não foram obtidos do Spotify diretamente, mas de arquivos disponíveis no Kaggle estando disponíveis nos seguintes endereços:

- DATASET: Spotify Dataset 1922-2021, ~600k Tracks

  URL: https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks

  Este dataset possui informações como nome de música, artistas, popularidade, além de informações de propriedades de músicas (energy, acoustiness, speechiness, valence etc.) que foram utilizadas para montagem do vetor de características para busca por similaridade.

- DATASET: Spotify Top 200 Songs

  URL: https://www.kaggle.com/dhruvildave/spotify-charts

  Este dataset contém informações sobre a quantidade de streams diários realizados das “top 200” músicas em diversos países.

- DATASET: Spotify Tracks DB

  URL: https://www.kaggle.com/zaheenhamidani/ultimate-spotify-tracks-db

Por fim, este dataset contém gêneros musicais associados às duas respectivas canções.

Embora os datasets sejam oriundos da API do Spotify, eles foram construídos por autores diferentes que utilizaram critérios diferentes sobre o que queriam dispor nos arquivos. Assim, o primeiro dataset da lista anteriormente exposta tem muitas informações importantes, mas não possui o gênero musical, bem como não possui a quantidade de streams por um dado período, ou o país onde ela foi popular. Assim, a ideia foi aproveitar bases que tivessem um ponto de junção comum, que foi a id da música.
Também é importante destacar que as bases não estarão normalizadas no sentido de todas as músicas estarem presentes nos arquivos obtidos. Isso se deve ao critério que os autores utilizaram para dispor os dados. Assim, no momento de cruzar as informações, os dados no Data Warehouse ficaram bem restritos no que se refere à quantidade de músicas.
Por fim, ainda que a restrição da quantidade de música seja fato, o processo de ETL mostrou que a base possui tamanho considerável. Assim, preferimos, para efeitos práticos do projeto, reduzir mais ainda o escopo do trabalho para um período compreendido entre 2018 e 2020 e os países sendo Brasil e Argentina.

DISCIPLINA: PROCESSAMENTO ANALÍTICO DE DADOS

Tema: construir um Data Warehouse (DW), descrever suas características e construir consultas com base nos dados.

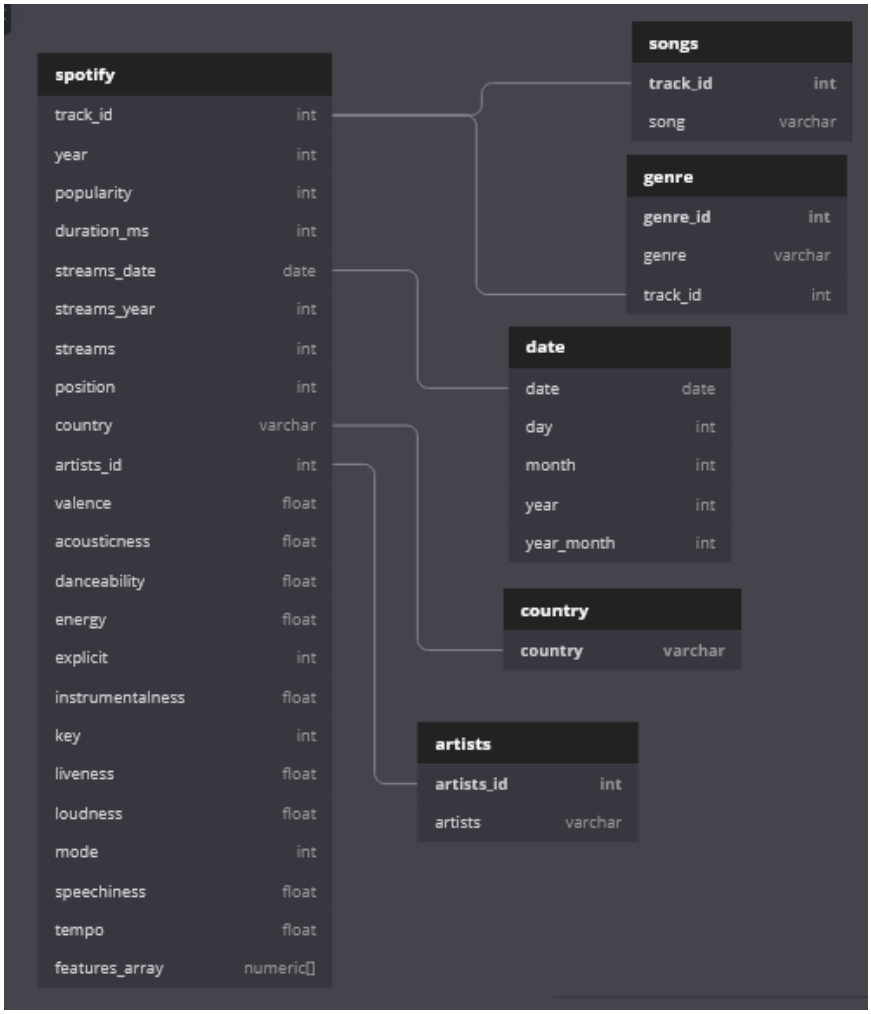
Contexto: informações sobre músicas do spotify.

PROJETO LÓGICO

Nosso projeto contou com 4 dimensões (songs, genre, country e artists).

A tabela de **fatos**, além das informações que caracterizam uma música para a montagem do vetor de características, possui as medidas a serem agregadas no DW, sendo elas *popularity* e *streams*, que querem dizer respectivamente o quanto uma música é popular (escala de 0 a 100), e a quantidade de *streams* diários realizados.

Na **dimensão** *date*, temos, a partir das informações diárias, uma **hierarquia** identificada nas grandezas de tempo, sendo *year*, *month* e *day*, em ordem de maior para o menor nível.



CONSULTAS

Nossos grupos de queries procurou trabalhar os seguintes pontos:

Q1. Observar os números de popularidade (*popularity*) de músicas (*songs*) e gêneros musicais (*genre*).

Q2. Observar os números de popularidade dos artistas (*artists*), considerando todas as suas músicas.

Q3. Observar a quantidade de *streams* realizadas em determinado período de tempo (*date*) e em qual localidade (*country*).

Q4. Conjunto de queries relacionadas a streams com visão mais ampla das dimensões envolvidas.

Como a tabela de **fatos** (*spotify)* possui granularidade alta por ter dados diários, a redundância precisou se trabalhada a fim de não obtermos agregações com resultados errados. Assim, optamos por criar um grupo de *Views Materializadas* que removem repetições e realizam algumas agregações. Os grupos Q1 e Q2 fazem uso dessas *views*. São elas:

mview_popularidade: une a tabela de fatos e as dimensões *artists*, *songs* e *country*, eliminando a redundância por conta da granularidade diária, que não será importante para os grupos Q1 e Q2.

```
CREATE MATERIALIZED VIEW mview_popularidade AS
    SELECT DISTINCT s.track_id, so.song, a.artists_id, a.artists,
        d.year, c.country, s.popularity
    FROM spotify s, artists a, songs so, date d, country c
    WHERE 1 = 1
    AND s.artists_id = a.artists_id
    AND s.track_id = so.track_id
    AND s.streams_date = d.date
    AND s.country = c.country;
```

Alguns resultados da View:

	track_id character varying	song character varying	artists_id integer	artists character varying	year integer	country character varying	popularity integer
1	017PF4Q3I4DBUiWoXk...	Break My Heart	1164	['Dua Lipa']	2020	Argentina	85
2	017PF4Q3I4DBUiWoXk...	Break My Heart	1164	['Dua Lipa']	2020	Brazil	85
3	01dmH2IPkrLNWIMPN...	Paradise	3823	['BTS']	2018	Argentina	67
4	01dmH2IPkrLNWIMPN...	Paradise	3823	['BTS']	2018	Brazil	67
5	01TnVDiet1DFTsyWKU...	NUMB	173	['XXXTENTACION']	2018	Argentina	76
6	01TnVDiet1DFTsyWKU...	NUMB	173	['XXXTENTACION']	2018	Brazil	76
7	01vv2AjxgP4uUyb8wa...	Morph	3222	['Twenty One Pilots']	2018	Argentina	67
8	01vv2AjxgP4uUyb8wa...	Morph	3222	['Twenty One Pilots']	2018	Brazil	67
9	03iCbZaM4OkRR4We6...	IDOL	4346	['BTS', 'Nicki Minaj']	2018	Brazil	69
10	03mMSLEJCPoGJwiQb...	Treat People With Kind...	2062	['Harry Styles']	2019	Argentina	72

mview_popularidade_media_artista e *mview_popularidade_media_musica*: fazem uso da *view* acima e realizam agregações por país, por ano, fornecendo uma média de popularidade para artistas e músicas, respectivamente.

```sql
CREATE MATERIALIZED VIEW mview_popularidade_media_artista AS
    SELECT artists_id, artists, country, year, avg(popularity) media_pop
    FROM mview_popularidade mp
    WHERE 1 = 1
    GROUP BY artists_id, artists, country, year
    ORDER BY artists, country, year

CREATE MATERIALIZED VIEW mview_popularidade_media_musica AS
    SELECT mp.track_id, mp.song, g.genre, mp.country, mp.year, avg(popularity) media_pop
    FROM mview_popularidade mp, genre g
    WHERE 1 = 1
    AND mp.track_id = g.track_id
    GROUP BY mp.track_id, mp.song, g.genre, mp.country, mp.year
    ORDER BY mp.country, mp.year, mp.track_id, mp.song, g.genre
```

Resultado da view *mview_popularidade_media_artista*:

artists_id integer	artists character varying	country character varying	year integer	media_pop numeric
401	['24kGoldn', 'iann dior']	Argentina	2020	99.0000000000000000
401	['24kGoldn', 'iann dior']	Brazil	2020	99.0000000000000000
526	['24kGoldn', 'Justin Bieber...	Argentina	2020	86.0000000000000000
1284	['5 Seconds of Summer']	Argentina	2018	75.0000000000000000
63	['5 Seconds of Summer']	Argentina	2018	82.0000000000000000
63	['5 Seconds of Summer']	Brazil	2018	82.0000000000000000
1284	['5 Seconds of Summer']	Brazil	2018	75.0000000000000000
2501	['6ix9ine', 'Anuel AA']	Argentina	2019	73.0000000000000000
1785	['6ix9ine']	Argentina	2020	80.0000000000000000
3007	['Agust D']	Argentina	2020	78.0000000000000000

Resultado da view *mview_popularidade_media_musica*:

track_id character varying	song character varying	genre character varying	country character varying	year integer	media_pop numeric
01dmH2IPkrLNWIMPNXI...	Paradise	Pop	Argentina	2018	67.0000000000000000
01TnVDiet1DFTsyWKUKo...	NUMB	Rap	Argentina	2018	76.0000000000000000
01vv2AjxgP4uUyb8waYO...	Morph	Rock	Argentina	2018	67.0000000000000000
08bNPGLD8AhKpnnERrA...	FRIENDS	Pop	Argentina	2018	81.0000000000000000
09IStsImFySgyp0pIQdqAc	The Middle	Dance	Argentina	2018	82.0000000000000000
09IStsImFySgyp0pIQdqAc	The Middle	Pop	Argentina	2018	82.0000000000000000
0b9oOr2ZgvyQu88wzixux9	This Is America	Hip-Hop	Argentina	2018	75.0000000000000000
0b9oOr2ZgvyQu88wzixux9	This Is America	Pop	Argentina	2018	75.0000000000000000
0b9oOr2ZgvyQu88wzixux9	This Is America	Rap	Argentina	2018	75.0000000000000000
0d2iYfpKoM0OCKvcLCk	Eastside (with Halsey & K	Dance	Argentina	2018	80.0000000000000000

Grupo Q1

Q1.1. Popularidade média das músicas no em todos os anos (2018 a 2020) em todos os países (com Brasil e Argentina na base), fazendo uso da view materializada com agregação.

```sql
SELECT song, country, avg(media_pop)
FROM mview_popularidade_media_musica
GROUP BY song, country
ORDER BY country, song
```

Resultado:

song character varying	country character varying	avg numeric
Without Me	Argentina	83.0000000000000000
X (feat. Maluma & Ozu...	Argentina	70.0000000000000000
Ya No Tiene Novio	Argentina	71.0000000000000000
Youngblood	Argentina	82.0000000000000000
1, 2, 3 (feat. Jason Der...	Brazil	72.0000000000000000
134340	Brazil	66.0000000000000000
365	Brazil	71.0000000000000000
Airplane pt.2	Brazil	66.0000000000000000
All The Stars (with SZA)	Brazil	79.0000000000000000

Observação: os valores para os dados presentes na base não se alteraram com a agregação por média, pelo fato de uma mesma música, em um mesmo país (até mesmo em países diferentes) em períodos de tempo (ano) diferentes manterem a mesmo valor de popularidade.

Q1.2. Popularidade média dos gêneros Rap e Pop no entre 2018 a 2020 em todos os países (com Brasil e Argentina na base), com uso de view materializada com agregação.

```
SELECT genre, avg(media_pop)
FROM mview_popularidade_media_musica
WHERE 1 = 1
AND genre IN ('Pop', 'Rap')
AND year BETWEEN 2018 AND 2020
GROUP BY genre;
```

Resultado:

genre character varying	avg numeric
Pop	75.1283018867924528
Rap	78.8783783783783784

Grupo Q2

q2.1. Observar a popularidade média das músicas de cada artistas em todos os países no ano de 2020, com uso de view materializada.

```
SELECT artists, country, media_pop
FROM mview_popularidade_media_artista
WHERE 1 = 1
AND year = 2020
```

Resultado

artists character varying	country character varying	media_pop numeric
['Taylor Swift']	Brazil	74.0000000000000000
['Taylor Swift']	Brazil	76.0000000000000000
['Taylor Swift']	Brazil	77.0000000000000000
['Taylor Swift']	Brazil	72.0000000000000000
['Taylor Swift']	Brazil	74.0000000000000000
['Taylor Swift']	Brazil	80.0000000000000000
['THE SCOTTS', 'Travis ...	Argentina	85.0000000000000000
['THE SCOTTS', 'Travis ...	Brazil	85.0000000000000000
['The Weeknd']	Argentina	80.0000000000000000

Q2.2. Popularidade média dos artistas e suas músicas no Brasil no ano de 2020, com uso de view materializada.

```
SELECT mp.artists, avg(mp.popularity) media_pop
FROM mview_popularidade mp
WHERE 1 = 1
AND mp.year = 2020
AND mp.country = 'Brazil'
GROUP BY mp.artists
ORDER BY mp.artists
```

Resultado

artists character varying	media_pop numeric
['Ariana Grande', 'Ty Dol...	84.0000000000000000
['Ariana Grande']	84.6923076923076923
['Arizona Zervas']	87.0000000000000000
['Ashnikko']	90.0000000000000000
['Bad Bunny', 'Jhay Cort...	100.0000000000000000
['BENEE', 'Gus Dappert...	84.0000000000000000
['Beyoncé']	69.0000000000000000
['Billie Eilish', 'Khalid']	89.0000000000000000
['Billie Eilish']	86.6666666666666667

Grupo Q3

Q3.1. Músicas com sua quantidade de streams realizados por país no ano de 2019.

```
SELECT so.song, c.country, d.year, sum(s.streams)
FROM country c, spotify s, date d, songs so
WHERE 1 = 1
AND s.country = c.country
AND s.track_id = so.track_id
AND s.streams_date = d.date
AND d.year = 2019
GROUP BY so.song, c.country, d.year
ORDER BY sum(s.streams) DESC;
```

Resultado:

song character varying	country character varying	year integer	sum bigint
Shallow	Brazil	2019	34114604
Calma - Remix	Argentina	2019	30126669
Señorita	Brazil	2019	30120819
Con Calma	Argentina	2019	29107635
7 rings	Brazil	2019	25821846
Otro Trago	Argentina	2019	25452937
China	Argentina	2019	24656662
Happier	Brazil	2019	22982100
Callaita	Argentina	2019	22840568
Secreto	Argentina	2019	22555982

Q3.2. Quantidade de streams realizados das músicas do gênero Rock no Brasil no ano de 2018.

```
SELECT so.song, d.year, sum(s.streams)
FROM country c, spotify s, date d, songs so,
    genre g
WHERE 1 = 1
AND s.country = c.country
AND s.track_id = so.track_id
AND s.streams_date = d.date
AND s.track_id = g.track_id
AND d.year = 2018
AND c.country = 'Brazil'
AND g.genre = 'Rock'
GROUP BY so.song, d.year
ORDER BY sum(s.streams) DESC;
```

Resutlado:

song character varying	year integer	sum bigint
Bad Liar	2018	1977541
Natural	2018	1591263
New Light	2018	1363047
Morph	2018	77687
My Blood	2018	73741
Chlorine	2018	68193

Q3.3. Média diária de streams realizados entre 2018 e 2020, por música, por país.

```
SELECT so.song, c.country, d.year, avg(s.streams)
FROM spotify s, songs so, date d, country c
WHERE 1 = 1
AND s.track_id = so.track_id
AND s.streams_date = d.date
AND s.country = c.country
AND d.year BETWEEN 2018 AND 2020
GROUP BY so.song, c.country, d.year
ORDER BY so.song, c.country, d.year
```

Reasultado

song character varying	country character varying	year integer	avg numeric
¿Quien Tu Eres?	Argentina	2018	24405.500000000000
+Linda	Argentina	2020	61851.051546391753
<3	Argentina	2020	31372.666666666667
1, 2, 3 (feat. Jason Derulo...	Argentina	2018	81774.112359550562
1, 2, 3 (feat. Jason Derulo...	Argentina	2019	27776.985294117647
1, 2, 3 (feat. Jason Derulo...	Brazil	2018	56221.428571428571
10,000 Hours (with Justi...	Brazil	2019	89401.000000000000
105 F Remix	Argentina	2019	32644.974358974359
11 PM	Argentina	2019	125076.566929133858
11 PM	Argentina	2020	36859.061224489796

Grupo Q4

Q4.1. Quantidade de streams realizados em cada país entre 2018 e 2020, considerando música, genero, artistas, ano.

```
SELECT so.song, g.genre, a.artists, d.year, c.country, sum(s.streams)
FROM spotify s, songs so, genre g, artists a, date d, country c
WHERE 1 = 1
AND s.track_id = so.track_id
AND s.track_id = g.track_id
AND s.artists_id = a.artists_id
AND s.streams_date = d.date
AND s.country = c.country
AND d.year BETWEEN 2018 AND 2020
GROUP BY so.song, g.genre, a.artists, d.year, c.country
ORDER BY so.song, d.year, c.country, g.genre, a.artists
```

Resultado

song character varying	genre character varying	artists character varying	year integer	country character varying	sum bigint
¿Quien Tu Eres?	Reggaeton	['Bad Bunny']	2018	Argentina	48811
1, 2, 3 (feat. Jason Der...	Dance	['Sofia Reyes', 'Jason D...	2018	Argentina	14555792
1, 2, 3 (feat. Jason Der...	Pop	['Sofia Reyes', 'Jason D...	2018	Argentina	14555792
1, 2, 3 (feat. Jason Der...	Dance	['Sofia Reyes', 'Jason D...	2018	Brazil	393550
1, 2, 3 (feat. Jason Der...	Pop	['Sofia Reyes', 'Jason D...	2018	Brazil	393550
1, 2, 3 (feat. Jason Der...	Dance	['Sofia Reyes', 'Jason D...	2019	Argentina	1888835
1, 2, 3 (feat. Jason Der...	Pop	['Sofia Reyes', 'Jason D...	2019	Argentina	1888835
134340	Pop	['BTS']	2018	Argentina	40330
134340	Pop	['BTS']	2018	Brazil	69191
200 Mph	Reggaeton	['Bad Bunny', 'Diplo', 'Nit...	2018	Argentina	53651

Q4.2. Adequação da query anterior para obter os streams realizados em outubro de 2018

```
SELECT so.song, g.genre, a.artists, c.country, sum(s.streams)
FROM spotify s, songs so, genre g, artists a, date d, country c
WHERE 1 = 1
AND s.track_id = so.track_id
AND s.track_id = g.track_id
AND s.artists_id = a.artists_id
AND s.streams_date = d.date
AND s.country = c.country
AND d.year_month = 201810 --outubro
GROUP BY so.song, g.genre, a.artists,  c.country
ORDER BY so.song, c.country, g.genre, a.artists
```

Resultado:

song character varying	genre character varying	artists character varying	country character varying	sum bigint
1, 2, 3 (feat. Jason Der...	Dance	['Sofia Reyes', 'Jason D...	Argentina	869136
1, 2, 3 (feat. Jason Der...	Pop	['Sofia Reyes', 'Jason D...	Argentina	869136
Always Remember Us ...	Dance	['Lady Gaga']	Brazil	157898
Always Remember Us ...	Pop	['Lady Gaga']	Brazil	157898
Arms Around You (feat...	Hip-Hop	['XXXTENTACION', 'Lil ...	Argentina	119244
Arms Around You (feat...	Rap	['XXXTENTACION', 'Lil ...	Argentina	119244
Arms Around You (feat...	Hip-Hop	['XXXTENTACION', 'Lil ...	Brazil	293129
Arms Around You (feat...	Rap	['XXXTENTACION', 'Lil ...	Brazil	293129
Better	Pop	['Khalid']	Brazil	127684
Better Now	Pop	['Post Malone']	Brazil	310088

DISCIPLINA: RECUPERAÇÃO DE INFORM. C/ BASE EM CONTEÚDO

Requisitos

Descrever como armazenar as imagens (ou outros dados complexos) no data warehouse projetado e implementado;

Armazenar os vetores de características no data warehouse mantido no PostgreSQL;

Inserção das features por meio da inserção das colunas em lista em nova coluna do dataframe

```
df_spotify['features_array'] = df_spotify[['valence', 'acousticness', 'danceability','energy' ,'explicit' , \
                                           'instrumentalness', 'key', 'liveness','loudness' , \
                                           'mode','speechiness' ]].values.tolist()
df_spotify
```

:

track_id	instrumentalness	key	...	popularity	speechiness	tempo	id_artists	streams_date	streams_year	streams	position	country	features_array
/k8hdRR	0.000000	6	...	89	0.2000	83.903	2	09/12/2019	2019.0	96687.0	155.0	Brazil	[0.218, 0.349, 0.511, 0.5660000000000001, 1, 0...
/k8hdRR	0.000000	6	...	89	0.2000	83.903	2	10/12/2019	2019.0	88501.0	183.0	Brazil	[0.218, 0.349, 0.511, 0.5660000000000001, 1, 0...
jAADbn4	0.000000	4	...	89	0.0333	115.284	5	24/05/2018	2018.0	52373.0	198.0	Brazil	[0.12, 0.934, 0.35100000000000003, 0.296000000...
jAADbn4	0.000000	4	...	89	0.0333	115.284	5	28/05/2018	2018.0	53482.0	175.0	Brazil	[0.12, 0.934, 0.35100000000000003, 0.296000000...

Criar consultas por similaridade range query e kNN (aceita-se o valor fixo para k, ex: os 5 mais similares) sobre o data warehouse que mantém os vetores de características.

Consultas por similaridade - kNN:

CREATE extension cube

SELECT id,cube(features_array) <-> cube(array[-array numérica de features-]) as distance

FROM spotify

ORDER BY cube(features_array) <-> cube(array[-array numérica de features-]) LIMIT 5;

Consultas por similaridade - RangeQuery:

---Function l2

CREATE or replace function l2(elem1 float[], elem2 float[]) returns float as $$

declare

size integer;

```
 somat float;
begin
 size := cardinality(elem1);
 somat := 0;
 for i in 1..size loop
    somat := somat + (ABS(elem1[i] - elem2[i])*ABS(elem1[i] - elem2[i]));
 end loop;
 return sqrt(somat);
end $$
language plpgsql;
--------------------------------------------------------------------------------
--RangeQuery
CREATE or replace function RangeQueryl2(qc float[], radius float)
 returns table (id integer, distance float) as $$
 begin
  return query
      select spotify.id, l2(features,qc) as distance
      from spotify
      where l2(features,qc) <= radius;

 end $$
 language plpgsql;
--------------------------------------------------------------------------------
TABLE spotify
SELECT * from RangeQueryl2(array[.........], 1.0);
```

Encontrar as músicas mais ouvidas:

```
SELECT * FROM public.spotify
```

ORDER BY streams desc;

Query Editor | Query History

```
1  SELECT * FROM spotify
2  ORDER BY streams desc;
```

Data Output | Explain | Messages | Notifications

	track_id character varying	popularity integer	year integer	artists_id integer	valence double precision
1	7e1VoRlNumuEkXlITM...	85	2020	1800	
2	24ySl2hOPGCDcxBxFl...	86	2020	579	0.6459999999
3	7e1VoRlNumuEkXlITM...	85	2020	1800	
4	1lWNylpZ477glVUDpJ...	76	2020	4232	
5	7ju97lgwC2rKQ6wwsf...	82	2020	1621	0.6459999999
6	7e1VoRlNumuEkXlITM...	85	2020	1800	
7	3PflrDoz19wz7qK7tYe...	84	2020	1741	
8	3PflrDoz19wz7qK7tYe...	84	2020	1741	
9	6ql0MU175Dk2DeoUjlr...	78	2020	3464	
10	7e1VoRlNumuEkXlITM...	85	2020	1800	

1º lugar:

7e1VoRlNumuEkXlITMHKLE

Query Editor | Query History

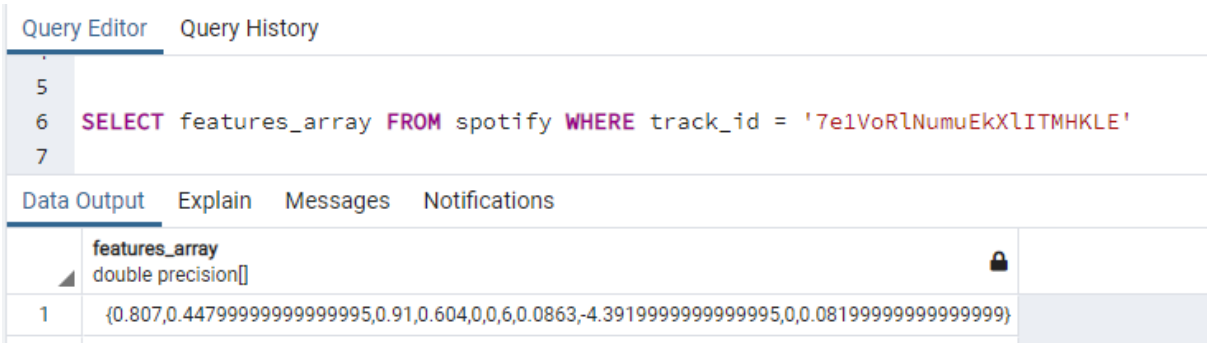
```
6  SELECT * FROM songs WHERE track_id = '7e1VoRlNumuEkXlITMHKLE'
7
8
9
```

Data Output | Explain | Messages | Notifications

	track_id [PK] character varying	song character varying
1	7e1VoRlNumuEkXlITMHKLE	Me Gusta (with Cardi B & Myke Towers)

Consulta de *features* da música com valor mais alto de streams:

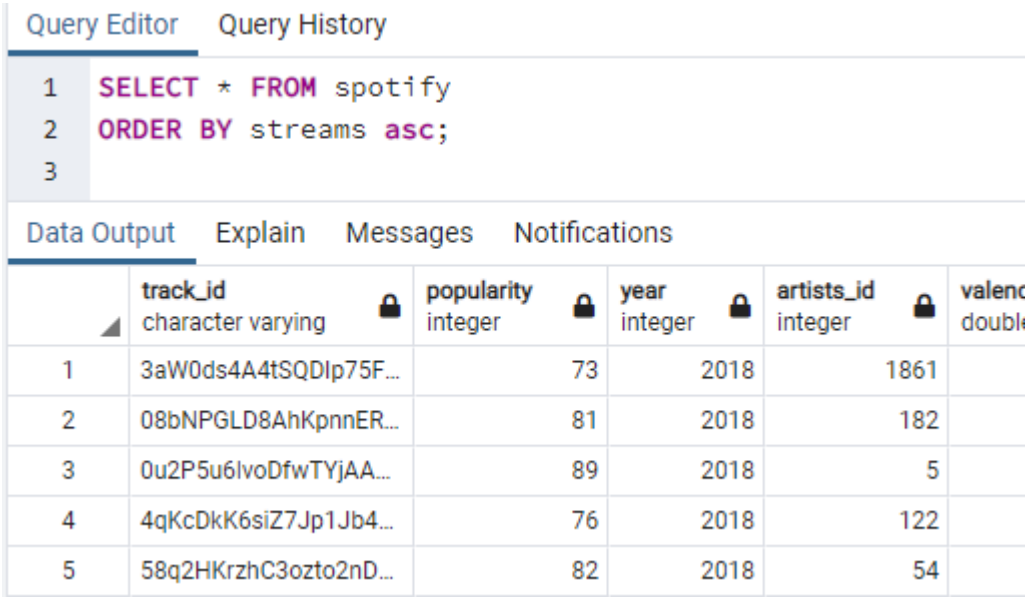
SELECT features_array FROM spotify WHERE track_id = '7e1VoRlNumuEkXlITMHKLE'



Consulta de músicas com valores mais baixos de streams:

SELECT * FROM spotify

ORDER BY streams asc;



	track_id character varying	popularity integer	year integer	artists_id integer	valenc doubl
1	3aW0ds4A4tSQDIp75F...	73	2018	1861	
2	08bNPGLD8AhKpnnER...	81	2018	182	
3	0u2P5u6lvoDfwTYjAA...	89	2018	5	
4	4qKcDkK6siZ7Jp1Jb4...	76	2018	122	
5	58q2HKrzhC3ozto2nD...	82	2018	54	

Consulta do gênero:

SELECT genre FROM genre WHERE track_id = '3aW0ds4A4tSQDIp75FqWTo'

```
3  SELECT genre FROM genre WHERE track_id = '3aW0ds4A4tSQDIp75FqWTo'
4
```

Data Output | Explain | Messages | Notifications

	genre character varying
1	Pop

***Feature* da música com menor valor de streams:**

SELECT features_array FROM spotify WHERE track_id = '3aW0ds4A4tSQDIp75FqWTo'

```
8  SELECT features_array FROM spotify WHERE track_id = '3aW0ds4A4tSQDIp75FqWTo'
```

Data Output | Explain | Messages | Notifications

	features_array double precision[]
1	{0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372}

{0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372}

Consulta de distância

SELECT track_id,cube(features_array) <-> cube(array[0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372]) as distance

FROM spotify

ORDER BY cube(features_array) <-> cube(array[0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372]) desc LIMIT 10;

```
36  SELECT track_id,cube(features_array) <-> cube(array[0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372]) as distance
37  FROM spotify
38  ORDER BY cube(features_array) <-> cube(array[0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372]) desc LIMIT 10;
```

Data Output | Explain | Messages | Notifications

	track_id character varying	distance double precision
1	5tyMJIMqaggzvuX7TtlrTe	13.992109112436337
2	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
3	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
4	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
5	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
6	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
7	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
8	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
9	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858
10	4ZLzoOkj0MPWrTLvooluaa	12.103335499724858

Consulta de gênero de música mais “distante”:

SELECT * FROM songs

WHERE track_id = '5tyMJlMqaggzvuX7TtlrTe'

```
41  SELECT * FROM songs
42  WHERE track_id = '5tyMJlMqaggzvuX7TtlrTe'
43
```

Data Output | Explain | Messages | Notifications

	track_id [PK] character varying	song character varying
1	5tyMJlMqaggzvuX7TtlrTe	Love song

Consulta por RangeQuery, raio = 10:

TABLE spotify

SELECT * FROM RangeQueryl2(array[0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372], 10) ORDER by distance DESC;

```
46  table spotify
47  SELECT * FROM RangeQueryl2(array[0.344,0.00267,0.557,0.7190000000000001,0,0,2,0.306,-4.515,0,0.0372], 10) ORDER by distance DESC;
```

Data Output | Explain | Messages | Notifications

	id character varying	distance double precision
1	60ynsPSSKe6O3sfwRnIBRf	9.925896290960328
2	60ynsPSSKe6O3sfwRnIBRf	9.925896290960328
3	60ynsPSSKe6O3sfwRnIBRf	9.925896290960328
4	60ynsPSSKe6O3sfwRnIBRf	9.925896290960328
5	60ynsPSSKe6O3sfwRnIBRf	9.925896290960328
6	60ynsPSSKe6O3sfwRnIBRf	9.925896290960328
7	60ynsPSSKe6O3sfwRnIBRf	9.925896290960328

DISCIPLINA: INTROCUÇÃO AO BIG DATA

Considerando o data warehouse construído (conforme especificado nos requisitos anteriores), porém sem levar em conta o armazenamento de vetores de características, o grupo deve discutir como ele poderia ser implantado em um NoSQL.

Breve panorama

Spotify tem sido um grande usuário do PostgreSQL desde o início das suas operações. Desde as primeiras versões do back-end do Spotify, tem sido o banco de dados relacional ideal e a escolha padrão para armazenamento persistente, quando funciona e atende às necessidades da operação. Pode-se dizer que PostgreSQL é um bom exemplo de tecnologia bastante madura. A menos que você sobrecarregue o Postgres, ele funciona sem ter que alterar ou compreender o código-fonte.

No início, o Spotify usou o PostgreSQL versão 8. Para lidar com as falhas do servidor, foram utilizados os recursos de replicação do banco de dados. Posteriormente surgiu o Postgres 9 e nesta versão a excelente replicação de streaming e funcionalidade de espera ativa. Um dos clusters de banco de dados mais importantes no Spotify, o cluster que armazena as credenciais do usuário (para login), é um cluster Postgres 9 (SPOTIFY R&D, 2013).

Qual é o NoSQL escolhido e por que da sua escolha?

Cassandra.

I. Um cluster Cassandra administrado adequadamente tem melhor replicação (especialmente operações de gravação);
II. Melhor comportamento na presença de problemas e falhas de rede, como partições ou falhas intermitentes.
III. Banco de dados distribuído, “Wide Column Store” com Apache License 2.0.
IV. Uso é recomendado como banco de dados OLAP (por exemplo, Data Warehouse) para lidar com grande volume de dados, e também é utilizado como banco de dados de série temporal;
V. Oferece escalabilidade horizontal linear e um dos bancos de dados mais escalonáveis com fragmentação automática (em termos de CAP, é AP (Disponível e Tolerante a Partições);
VI. Banco de dados descentralizado (leaderless) com replicação automática e replicação multi-datacenter, é tolerante a falhas;
VII. Possui linguagem de consulta amigável e semelhante a SQL: Cassandra Query Language (CQL);
VIII. É utilizado em aplicativos escaláveis da web, que precisam lidar com uma grande quantidade de operações de gravação e leitura com escalabilidade linear.

Além disto, por benchmarking do próprio Spotify, Instagram/Facebook (GU, 2018), e posição como NoSQL no ranking DB-Engines (2021).

371 systems in ranking, May 2021

Rank May 2021	Rank Apr 2021	Rank May 2020	DBMS	Database Model	Score May 2021	Score Apr 2021	Score May 2020
1.	1.	1.	Oracle	Relational, Multi-model	1269.94	-4.98	-75.50
2.	2.	2.	MySQL	Relational, Multi-model	1236.38	+15.69	-46.26
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	992.66	-15.30	-85.64
4.	4.	4.	PostgreSQL	Relational, Multi-model	559.25	+5.73	+44.45
5.	5.	5.	MongoDB	Document, Multi-model	481.01	+11.04	+42.02
6.	6.	6.	IBM Db2	Relational, Multi-model	166.66	+8.88	+4.02
7.	7.	↑ 8.	Redis	Key-value, Multi-model	162.17	+6.28	+18.69
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	155.35	+3.18	+6.23
9.	9.	9.	SQLite	Relational	126.69	+1.64	+3.66
10.	10.	10.	Microsoft Access	Relational	115.40	-1.33	-4.50
11.	11.	11.	Cassandra	Wide column	110.93	-3.92	-8.22

Fonte: https://db-engines.com/en/ranking

Qual é o modelo de dados do NoSQL escolhido?

NoSQL orientado a coluna (wide-column)

Por que seria interessante migrar do PostgreSQL ao NoSQL escolhido? detalhar a motivação.

Schema flexível

Tolerância a falhas

Disponibilidade

Como implantar as tabelas de dimensão e de fato no NoSQL escolhido?

Implementação em CQL:

```
'CREATE TABLE IF NOT EXISTS artists ('
    'artists_id int,'
    'artists varchar,'
    'CONSTRAINT PK_artists_id PRIMARY KEY(artists_id)'
')',

'CREATE TABLE IF NOT EXISTS genre ('
    'genre_id int,'
    'genre varchar,'
```

```
    'track_id varchar,'
    'CONSTRAINT PK_genre_id PRIMARY KEY(genre_id)'
')',

'CREATE TABLE IF NOT EXISTS year ('
    'year int,'
    'CONSTRAINT PK_year PRIMARY KEY(year)'
')',

'CREATE TABLE IF NOT EXISTS country ('
    'country varchar,'
    'CONSTRAINT PK_country_id PRIMARY KEY(country)'
')',

'CREATE TABLE IF NOT EXISTS songs ('
    'track_id varchar,'
    'song varchar,'
    'CONSTRAINT PK_track_id PRIMARY KEY(track_id)'
')',

'CREATE TABLE IF NOT EXISTS spotify ('
    'track_id varchar,'
    'popularity int,'
    'year int,'
    'artists_id int,'
    'valence float,'
    'acousticness float,'
    'danceability float,'
```

```
        'duration_ms int,'
        'energy float,'
        'explicit int,'
        'instrumentalness float,'
        'key int,'
        'liveness float,'
        'loudness float,'
        'mode int,'
        'speechiness float,'
        'tempo float,'

        'streams_date date,'
        'streams_year int,'
        'streams int,'
        'position int,'
        'country varchar,'

        'features_array number[]'
    ')',

    'CREATE INDEX idx_genre_track_id ON genre(track_id)',
)
```

Qual seria a sintaxe das consultas analíticas no NoSQL escolhido? apresente exemplos.

De forma semelhante ao PostgreSQL:

--Q1 - identificar a média de popularidade geral ou de gêneros músicais

--a fim de realizar alguma campanha de marketing sobre determinada preferência musical

--q1.1. Popularidade média das músicas entre os anos de 2018 e 2020

--Em todos os países (só tem o Brasil na base)

--Utilizando a view materializada com agregação

```
SELECT song, country, year, media_pop

FROM mview_popularidade_media_musica

WHERE 1 = 1

AND year = 2018;
```

Referências

DB-ENGINES. DB-Engines Ranking. 2021. Acessado em: < https://db-engines.com/en/ranking>. Acesso em: 02/05/2021.

GU, D. Open-sourcing a 10x reduction in Apache Cassandra tail latency. **Medium**. 2018. Acessado em: <https://instagram-engineering.com/open-sourcing-a-10x-reduction-in-apache-cassandra-tail-latency-d64f86b43589>. Acesso em: 02/05/2021.

SPOTIFY R&D. **In praise of "boring" technology**. 2013. Acessado em: <https://engineering.atspotify.com/2013/02/25/in-praise-of-boring-technology/>. Acesso em: 02/05/2021.