



Bacharelado em Ciência da Computação

Processamento Massivo de Dados

## **Fase Final**

**Tema: Recomendação de Música**

Profa. Dra. Sahudy Montenegro González

### **Grupo 8**

Thainá de Almeida Santana - RA: 761067

Vitor Pacheco de Carvalho - RA: 760941

Sorocaba - SP

27 de Setembro de 2022

## **Sumário**

<b>1. INTRODUÇÃO</b>	<b>3</b>
<b>2. FONTE DE DADOS</b>	<b>3</b>
<b>3. TECNOLOGIAS UTILIZADAS</b>	<b>5</b>
3.1 CouchDB	5
3.2 Neo4j	7
<b>4. FLUXO DOS DADOS</b>	<b>10</b>
<b>5. TRANSFORMAÇÕES</b>	<b>11</b>
<b>6. CONSULTAS</b>	<b>11</b>
6.1.1 CONSULTA 1	12
6.1.2 CONSULTA 2	14
<b>6.1.3 CONSULTA 3</b>	<b>16</b>
6.1.4 CONSULTA 4	19
<b>Imagem 33 : Resultado da consulta 4</b>	<b>20</b>
<b>7. CONCLUSÃO</b>	<b>20</b>
<b>8. BIBLIOGRAFIA</b>	<b>20</b>

## 1. INTRODUÇÃO

O projeto tem como objetivo indicar músicas com base em dados relacionados ao gosto do usuário, através de [pesquisas](#) vimos que a música além de ser algo muito importante culturalmente também é capaz de ajudar em algumas áreas da saúde, já que ativa várias partes do nosso cérebro.

Realizaremos uma aplicação ETL, que diz respeito a extração, transformação e carga de dados. Utilizaremos as tecnologias Apache Spark e Neo4j, suas utilizações serão explicadas no tópico [Tecnologias Utilizadas](#), para moldar e consultar os dados e gerar indicações de músicas e artistas para o usuário.

## 2. FONTE DE DADOS

Os dados que serão utilizados no trabalho são provenientes de dois datasets, o dataset com informações sobre músicas foi encontrado em um [repositório do github](#), que contém diversas informações sobre cada uma delas e todas tem seu lançamento e alta popularidade entre 2010 e 2019. Outro dataset que será utilizado foi gerado pelo site extendsclass, tendo dados relacionados ao usuário.

Mais detalhes sobre o dataset podem ser visualizados nas tabelas abaixo:







Colunas	Descrição
Título	Nome da música
Artista	Artista que cantou
Gênero	Gênero da música
Ano	Ano em que ficou no top 50 do Spotify
BPM	Batidas por minuto, ajuda a determinar o ritmo
Energia	A energia de uma música - quanto maior o valor, mais enérgico
Dançabilidade	Quanto maior o valor , mais fácil é dançar esta música
Volume	Quanto maior o valor, mais alta a música
Liveness	Quanto maior o valor, mais provável é que a música seja uma gravação ao vivo
Valência	Quanto maior o valor, mais o clima positivo para a música
Comprimento	A duração da música

Acústica	Quanto maior o valor, mais acústica a música é
Fala	Quanto maior o valor, mais palavras faladas a música contém
Popularidade	Quanto maior o valor, mais popular é a música

Tabela 1: Colunas do dataset de música

Coluna
id_usuario
Nome

Tabela 2: Colunas do dataset de usuário

	Field name	Data type	Setting
  	<input type="text" value="id"/>	<input type="text" value="Index"/>	From <input type="text" value="1"/>
  	<input type="text" value="nome"/>	<input type="text" value="First name"/>	

**Add field**

Format:  # Rows:  # Files:  Filename:

Delimiter:  Quote:  Escape:  End line:

Include header:

Imagem 1: Geração dos dados no site extendsclass

### 3. TECNOLOGIAS UTILIZADAS

Algumas tecnologias serão necessárias para o desenvolvimento do projeto, o Apache Spark foi escolhido para o tratamento e manipulação prévia dos dados junto com a linguagem de programação Python, deixando os dados que serão trabalhados mais coerentes. Serão utilizados dois bancos de dados NoSQL, o Neo4j que é orientado a grafo será usado para as consultas onde precisamos percorrer relações, seja entre amigos ou músicas. O segundo banco de dados escolhido para o projeto foi o CouchDB que é orientado a documentos, que facilita nas consultas baseadas em alguns campos de música e também é ideal para alta escalabilidade suportando grandes quantidades de músicas. Também foi utilizado a biblioteca Pandas para facilitar a criação de um arquivo csv que irá servir como fonte para os dados que serão guardados no CouchDB

#### 3.1 CouchDB

O principal objetivo da utilização do CouchDB neste projeto é a facilidade de fazer consultas baseadas em intervalo de valores, como uma das consultas propostas, e também para armazenar todas as colunas com as variadas propriedades de cada música.

Para implementar a conexão e operações básicas foi necessário fazer a instalação da biblioteca couchdb que auxilia no objetivo:

```
pip install couchdb
```

Imagem 2 : Código para instalação do CouchDB

Fizemos então a conexão com o CouchDB, onde importamos as bibliotecas csv e couchdb, mapeamos o server e demos um nome para o banco.

```
In [25]: #Conexão com Couchdb
import csv
import couchdb

couch = couchdb.Server('http://admin:123456789@127.0.0.1:5984')

dbname = "pmd22" #Nome do banco
```

Imagem 3: Conexão com o CouchDB

Caso o banco não exista, criamos então o banco de dados passando o nome definido anteriormente, carregamos o arquivo csv com as informações dos usuários contendo 2000 registros e então fizemos a inserção no CouchDB.

```
#Criação do Banco de Dados e Inserção dos dados da fonte
try:
    db = couch[dbname]
except:
    db = couch.create(dbname)
with open(r'B:\Faculdade\PMO\musicWithIds\music.csv', encoding="utf-8") as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    keys = next(csv_reader)
    line_count = 0
    for values in csv_reader:
        values[3] = int(values[3])if values[3].isdigit() else 0
        values[4] = int(values[4])if values[4].isdigit() else 0
        values[5] = int(values[5])if values[5].isdigit() else 0
        values[6] = int(values[6])if values[6].isdigit() else 0
        values[7] = int(values[7])if values[7].isdigit() else 0
        values[8] = int(values[8])if values[8].isdigit() else 0
        db.save(dict(zip(keys, values)))
```

Imagem 4: Criação do banco de dados e inserção dos dados da fonte

Para realizar um teste fizemos uma leitura de dados do CouchDB, onde é retornado os usuários com o campo nome igual a “Madalyn”.

```
In [27]: #Leitura de dados do Couchdb
for doc in db.find({
    'selector': {
        'nome': {
            '$eq': "Madalyn"
        }
    }
}):
    print(doc)

<Document '1'@'1-251939b6af72846e2f96e94dd0efe460' {'nome': 'Madalyn'}>
<Document '1158'@'1-251939b6af72846e2f96e94dd0efe460' {'nome': 'Madalyn'}>
<Document '153'@'1-251939b6af72846e2f96e94dd0efe460' {'nome': 'Madalyn'}>
<Document '428'@'1-251939b6af72846e2f96e94dd0efe460' {'nome': 'Madalyn'}>
<Document '528'@'1-251939b6af72846e2f96e94dd0efe460' {'nome': 'Madalyn'}>
<Document '561'@'1-251939b6af72846e2f96e94dd0efe460' {'nome': 'Madalyn'}>
```

Imagem 5: Leitura de dados do CouchDB através de consulta

Fazendo a comparação com a consulta diretamente no CouchDB é possível verificar que o resultado retornado corresponde.

pmd22 > Mango Query

Query history

Mango Query ?

```
1 {
2   "selector": {
3     "nome": {
4       "$eq": "Madalyn"
5     }
6   }
7 }
```

Run Query Explain manage indexes

Executed in 158 ms

Table {} JSON

	_id	nome
<input type="checkbox"/>	1	Madalyn
<input type="checkbox"/>	1158	Madalyn
<input type="checkbox"/>	153	Madalyn
<input type="checkbox"/>	428	Madalyn
<input type="checkbox"/>	528	Madalyn
<input type="checkbox"/>	561	Madalyn

Imagem 6: Mesma consulta realizada diretamente no CouchDB

### 3.2 Neo4j

Neste projeto o Neo4j foi utilizado visando as consultas baseadas em relações entre pessoas e outras pessoas, e pessoas e suas músicas favoritas. Sendo um banco baseado em grafos, essas consultas são mais eficientes e também trazem informações necessárias para o projeto

Iniciamos fazendo um teste para carregar os dados das músicas e usuários no Neo4j, para isso foi necessário acrescentarmos os arquivos .csv para dentro da pasta correta. Para verificar a pasta correta fomos no Neo4j, no menu do projeto em específico, open folder e import. (Imagem 7).

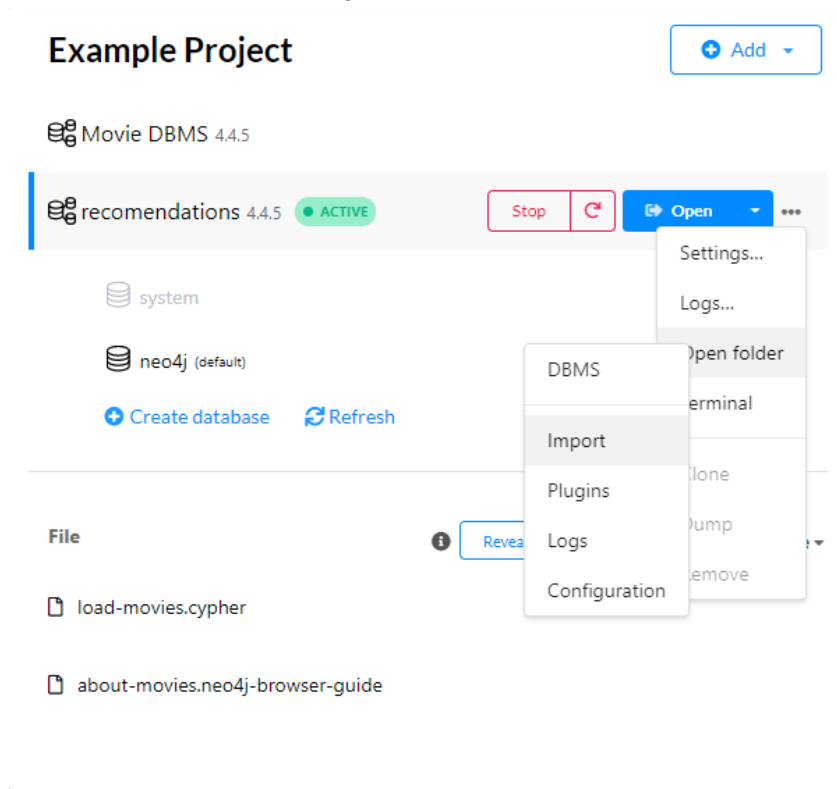


Imagem 7: Encontrando o caminho correto da pasta

Ao clicar em import é aberto a pasta correta e é possível copiar os arquivos desejados para dentro dela.

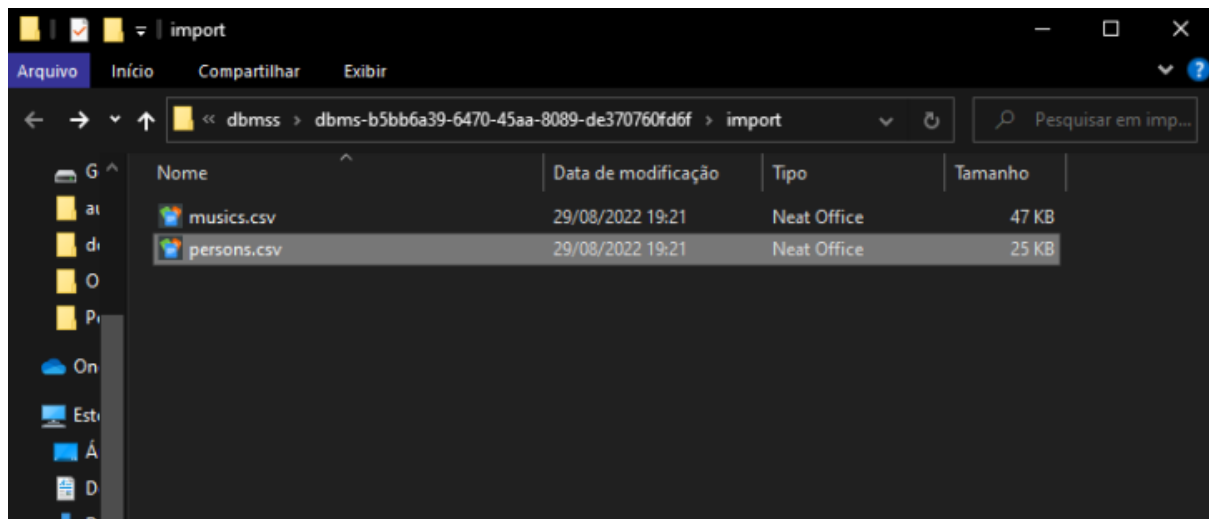


Imagem 8: Exibindo a pasta correta e os arquivos necessários para o projeto

Para carregar o arquivo de música executamos o seguinte comando:

```
LOAD CSV WITH HEADERS FROM "file:///music.csv" AS musicas
CREATE (m:musicas)
SET m = musicas,
    m.ano = toInteger(musicas.ano),
    m.BPM = toInteger(musicas.BPM),
    m.energia = toInteger(musicas.energia),
    m.dancabilidade = toInteger(musicas.dancabilidade),
    m.volume = toInteger(musicas.volume),
    m.liveness = toInteger(musicas.liveness),
    m.valencia = toInteger(musicas.valencia),
    m.comprimento = toInteger(musicas.comprimento),
    m.acustica = toInteger(musicas.acustica),
    m.fala = toInteger(musicas.fala),
    m.popularidade = toInteger(musicas.popularidade);
```

Imagem 9: Comando para ler arquivo csv e criar nós de música



Após carregar o arquivo fizemos o teste de uma consulta para retornar todos os nós de música, e o resultado foi este:

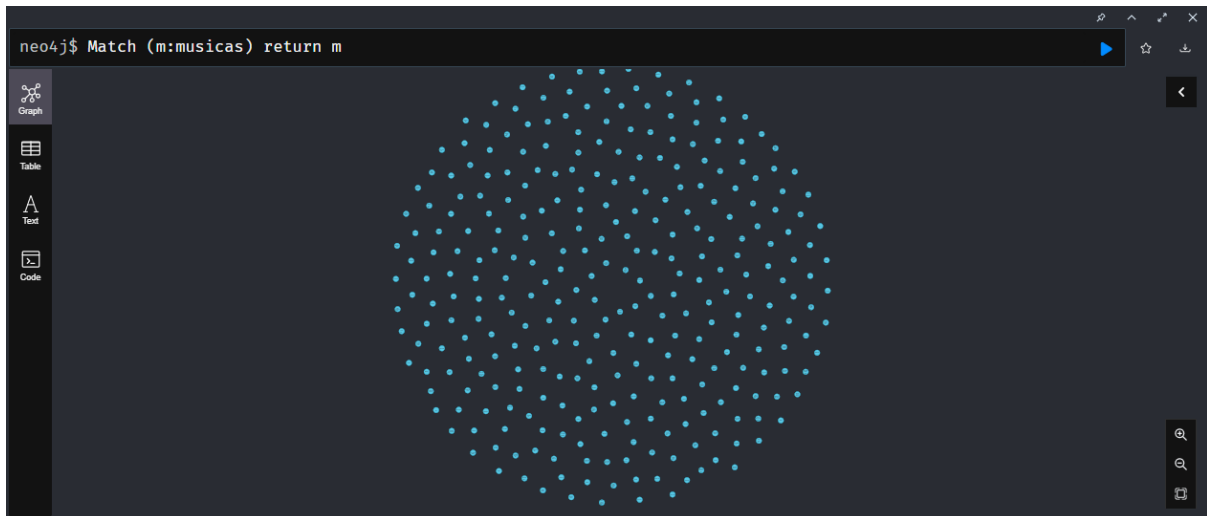


Imagem 10: Retorno da consulta para exibir nós de música

Em seguida fizemos os mesmos testes para o arquivo de usuários:

```
LOAD CSV WITH HEADERS FROM "file:///persons.csv" AS persons
CREATE (p:persons)
SET p = persons,
    p.id = toInteger(persons.id);
```

Imagem 11: Comando para ler arquivo csv e criar nós de usuários (persons)

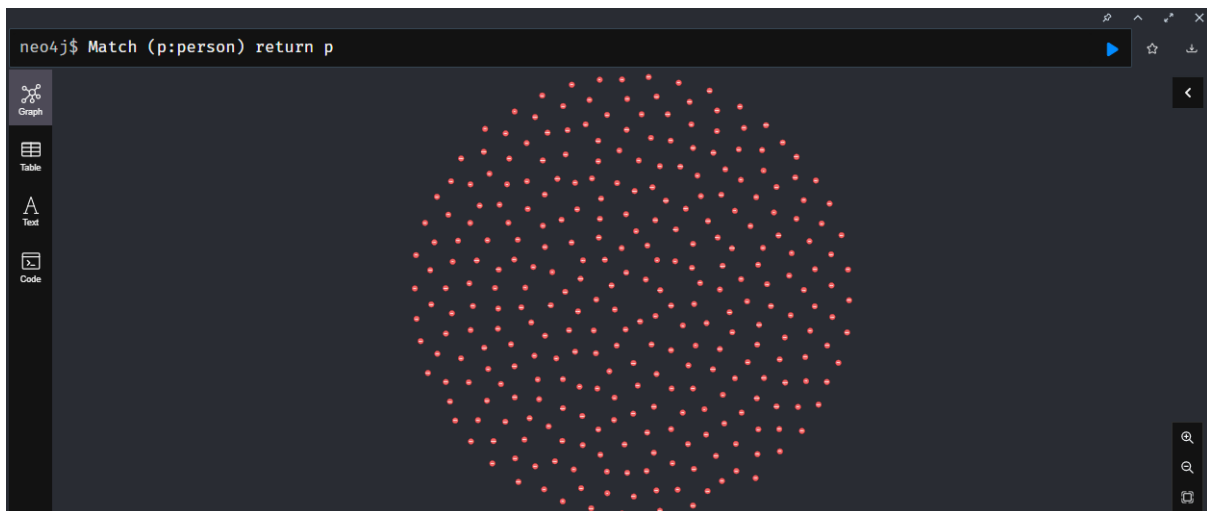


Imagem 12: Retorno da consulta para exibir nós de usuários (persons)

A estrutura que teremos no Neo4j ao final do projeto será semelhante a imagem a seguir:

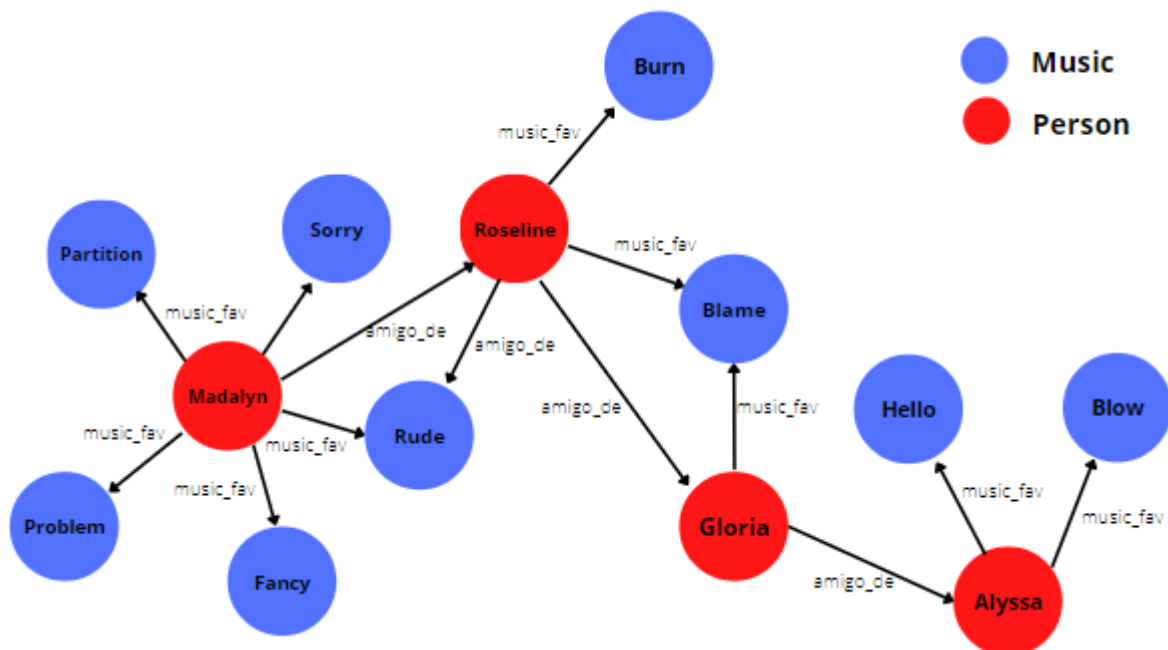


Imagem 13: Exemplo de fluxo no Neo4j

#### 4. FLUXO DOS DADOS

No fluxo dos dados é possível verificar toda a trajetória dos dados, desde sua origem, nos arquivos “music.csv” que contém as informações dos usuários e “person.csv” que contém as informações das músicas, passando pelo Spark onde é feito as transformações e salvando em um CSV utilizando Python quando necessário, até chegar no CouchDB e no Neo onde serão utilizados para as consultas.

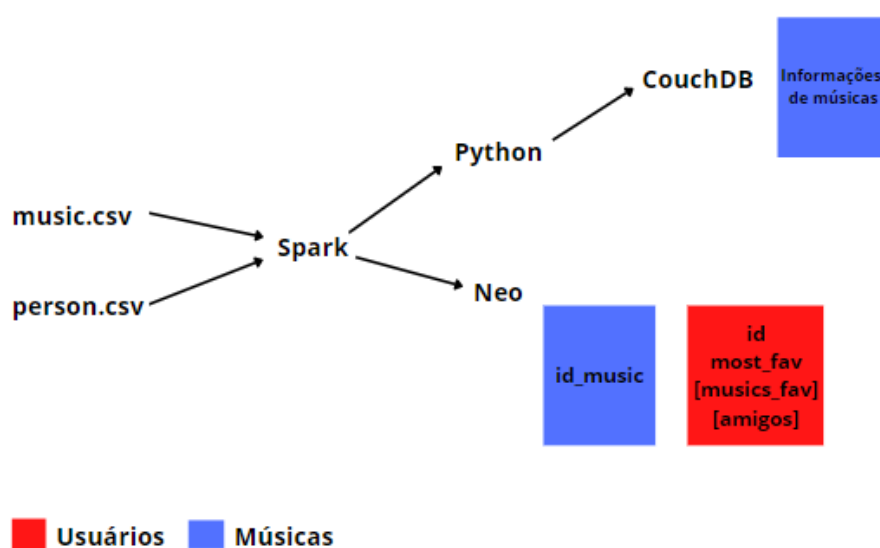


Imagem 14 : Fluxo dos processos sobre os dados

## 5. TRANSFORMAÇÕES

Nos arquivos coletados, serão feitas algumas transformações para tornar possível as consultas. No dataset relacionado as músicas será adicionado uma coluna 'Id\_music' para identificar cada música, e também será retirado as colunas 'Volume', 'Liveness', 'Acoustic', 'Fala' e 'Comprimento'.

Já no dataset de usuários será adicionado uma coluna 'music\_fav' que indicará três músicas favoritas do usuário, ela será preenchida com valores aleatórios que seguirá o intervalo dos id's do outro dataset. Além disso, neste dataset será incluído uma coluna 'amigos' que conterá inicialmente quatro id's de amigos do usuário.

Todas alterações que serão feitas são para diminuir o tamanho dos dados, já que não utilizaremos todas as informações e com isso otimizar espaço.

## 6. CONSULTAS

1. Retornar as músicas que tenham valores próximos de BPM, energia, dançabilidade e valência da música favorita do usuário.(Grafo + Documentos)
2. Retornar as músicas agrupadas pelos gêneros das músicas favoritas do usuário ordenadas por ano.(Grafo + Documentos)
3. Retornar uma playlist com músicas de pessoas que compartilhem a mesma música favorita, percorrendo cinco níveis de profundidade. (Grafo + Documento)
4. Retornar uma playlist levando em consideração as músicas favoritas de amigos em até três níveis de profundidade, ordenada pela popularidade. (Grafo + Documento)

### 6.1 CÓDIGO E TESTE DAS CONSULTAS

#### 6.1.1 CONSULTA 1

Esta consulta tem como objetivo retornar para o usuário músicas que são semelhantes a sua música favorita, para isso ela se baseia nos valores de BPM, energia, dançabilidade e valência. Primeiramente é pego no Neo4j a informação do id da música favorita do usuário.

São realizadas duas consultas no CouchDB, a primeira é para conseguir as características necessárias da música favorita do usuário e a segunda é para retornar todas as músicas em que essas características sejam próximas.

```

#Consulta 1 - Retornar as músicas que tenham valores próximos de BPM, energia, dançabilidade e valência da música favorita do usu
idClientC1 = '1534'
allDocs = []

#grafo
c1_mIds = spark.read\
.format("org.neo4j.spark.DataSource")\
.option("url", "neo4j://localhost:7687")\
.option("authentication.type","basic")\
.option("authentication.basic.username", "neo")\
.option("authentication.basic.password", "1234")\
.option("query","MATCH (p:Person) WHERE p.id_person = " + idClientC1 + " RETURN p.most_list")\
.load()

#Musica favorita do cliente
for doc in db.find({
  'selector': {
    "id_music": {
      "$eq": c1_mIds.collect()[0][0]
    }
  },
  "fields": ["BPM", "energia", "dancabilidade", "valencia","titulo", "id_music"]
}):
  fav = doc

print(fav["BPM"], fav["energia"], fav["dancabilidade"], fav["valencia"], fav["titulo"], "\n\n\n")

```

Imagem 15 : Códigos da consulta 1

```

#Consulta baseada no intervalo
for doc in db.find({
  "selector": {
    "$and": [
      {
        "BPM": {
          "$and": [
            {
              "$gte": int(fav["BPM"]) - 8
            },
            {
              "$lte": int(fav["BPM"]) + 8
            }
          ]
        }
      },
      {
        "energia": {
          "$and": [
            {
              "$gte": int(fav["energia"]) - 6
            },
            {
              "$lte": int(fav["energia"]) + 6
            }
          ]
        }
      }
    ]
  }
})

```

Imagem 16 : Códigos da consulta 1

```

{
  "dancabilidade": {
    "$and": [
      {
        "$gte": int(fav["dancabilidade"]) - 6
      },
      {
        "$lte": int(fav["dancabilidade"]) + 6
      }
    ]
  },
  "valencia": {
    "$and": [
      {
        "$gte": int(fav["valencia"]) - 8
      },
      {
        "$lte": int(fav["valencia"]) + 8
      }
    ]
  }
},
"limit": 200
}):
print(doc["BPM"], doc["energia"], doc["dancabilidade"], doc["valencia"], doc["titulo"])

```

Imagem 17: Códigos da consulta 1

No retorno da consulta no Neo4J, o id retornado está correto, pois a música favorita do usuário 3 é a música 52.

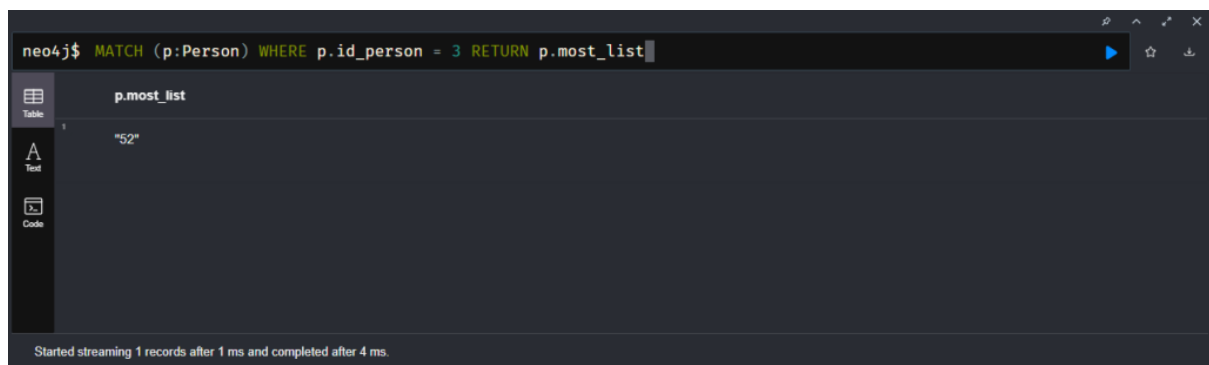


Imagem 18 : Teste da consulta 1 executado no Neo4j

Na imagem 19 é apresentado os dados da música favorita do usuário e das que foram retornadas por conta da sua proximidade, as informações estão dentro dos limites esperados.

```

125 94 60 55 We Are One (Ole Ola) [The Official 2014 FIFA World Cup Song]

118 88 65 49 Best Song Ever
131 93 66 53 Judas
128 93 57 58 Sweet Nothing (feat. Florence Welch)
124 93 63 47 This Is What You Came For
126 92 54 51 Under Control
125 94 60 55 We Are One (Ole Ola) [The Official 2014 FIFA World Cup Song]

```

Imagem 19 : Teste da consulta 1 após retorno do CouchDB

## 6.1.2 CONSULTA 2

A consulta número dois é baseada nos gêneros das três músicas favoritas do usuário. É retornada a informação de todas as músicas disponíveis em cada um dos gêneros, ordenadas por ano.

O usuário em questão da consulta é passado através da variável 'idClientC2', é utilizado também um vetor com os ID's retornados para que seja realizada as consultas no CouchDB em uma única chamada utilizando um 'OR'.

```
#Consulta 2 - Retornar as músicas agrupadas pelos gêneros das músicas favoritas do usuário ordenadas por ano.(Documentos)
idClientC2 = '657'
musicsFav = []
musicsgens = []

#Busca as músicas favorita do cliente
c1_mIds = spark.read\
    .format("org.neo4j.spark.DataSource")\
    .option("url", "neo4j://localhost:7687")\
    .option("authentication.type", "basic")\
    .option("authentication.basic.username", "neo")\
    .option("authentication.basic.password", "1234")\
    .option("query", "MATCH (p:Person) - [re:music_fav] -> (m:Music) WHERE p.id_person = " + idClientC2 + " RETURN m.id_musica")\
    .load()

for item in c1_mIds.collect():
    musicsFav.append(str(item[0]))

print(musicsFav)
#Genero da musica favorita do cliente
for doc in db.find({
    "selector": {
        "id_musica": {
            "$or": musicsFav
        }
    },
    "sort": [{"ano": "asc"}]
}):
    musicsgens.append(doc["genero"])
print(musicsgens)
```

Imagem 20 : Códigos da consulta 2

```
for doc in db.find({
    "selector": {
        "$or": [
            {
                "genero": {
                    "$or": musicsgens
                }
            }
        ]
    },
    "sort": [
        {
            "genero": "asc"
        },
        {
            "ano": "asc"
        }
    ]
},
    "limit": 603
}):
    print(doc["genero"], doc["ano"])
```

Imagem 21 : Códigos da consulta 2

Para teste foi utilizado a pessoa com ID 3 novamente, e verificamos as três músicas favoritas deste usuário.

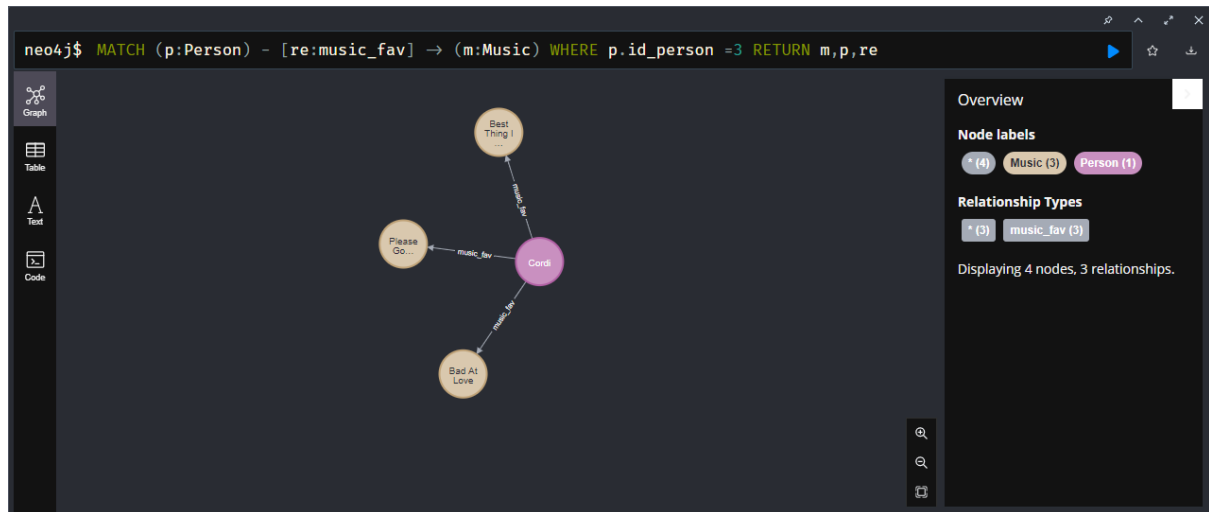


Imagem 22 : Teste da consulta 1 executado no Neo4j

É possível verificar que os ID's das músicas batem com os ID's que estão no vetor que é passado para o CouchDB.

The image shows the Neo4j web interface with a table view. The query bar contains: `neo4j$ MATCH (p:Person) - [re:music_fav] -> (m:Music) WHERE p.id_person =3 RETURN m.id_music`. The left sidebar has icons for Table, Text, and Code. The main area displays a table with the following data:

	m.id_music
1	39
2	52
3	391

At the bottom, a status message reads: 'Started streaming 3 records in less than 1 ms and completed after 2 ms.'

Imagem 23 : Consulta teste no Neo4j para verificação dos ID's

Na imagem 24 é possível verificar que o resultado retornado do CouchDB está correto, onde as músicas estão agrupadas por gênero e ordenadas por ano conforme esperado.

```
['577', '480', '8']
['art pop', 'permanent wave', 'dance pop']
art pop 2010
art pop 2013
art pop 2014
art pop 2015
art pop 2016
art pop 2017
art pop 2017
art pop 2017
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
dance pop 2010
```

Imagem 24 : Consulta teste no Neo4j para verificação dos ID's

### 6.1.3 CONSULTA 3

A consulta três tem como objetivo retornar as músicas de pessoas que gostem das mesmas músicas, assim é possível criar uma playlist para o usuário com músicas que tenham chances dele gostar.

O usuário foco da consulta é passado para a query através de uma variável chamada 'idClientC3'. O retorno da consulta é salvo em um dataset e posteriormente pegamos os ids e colocamos em um vetor, para que possamos realizar a consulta no CouchDB utilizando o operador "OR" ao invés de fazer uma consulta para cada id.

```
#Consulta 3 - Retornar uma playlist com músicas de pessoas que compartilhem a mesma música favorita, percorrendo cinco níveis de
idClientC3 = '45'
allIds = []

c3_mIds = spark.read\
.format("org.neo4j.spark.DataSource")\
.option("url", "neo4j://localhost:7687")\
.option("authentication.type", "basic")\
.option("authentication.basic.username", "neo")\
.option("authentication.basic.password", "1234")\
.option("query", "MATCH (p:Person) - [fav:music_fav] -> (m:Music) <- [mu:music_fav*1..5]\
- (p2:Person) - [mu2:music_fav] -> (demais_music:Music) \
WHERE m.id_music = toInteger(p.most_list) \
AND p.id_person = "+ idClientC3+" \
AND demais_music.id_music = toInteger(p2.most_list) \
AND p.id_person <> p2.id_person \
RETURN demais_music.id_music")\
.load()

for idmusic in c3_mIds.collect():
    allIds.append(str(idmusic[0]))

for doc in db.find({
    "selector": {
```

Imagem 25 : Códigos da consulta 3



```
"id_music": {
  "$on": allIds
},
},
"limit": 200
}):
print(doc["titulo"], doc["artista"], doc["genero"], doc["ano"])
```

Imagem 26 : Códigos da consulta 3

Para testar se o retorno da consulta no Neo4j estava correto utilizamos a mesma consulta exibindo todos os nós e relações e diminuimos a profundidade para 2 para uma melhor visualização.

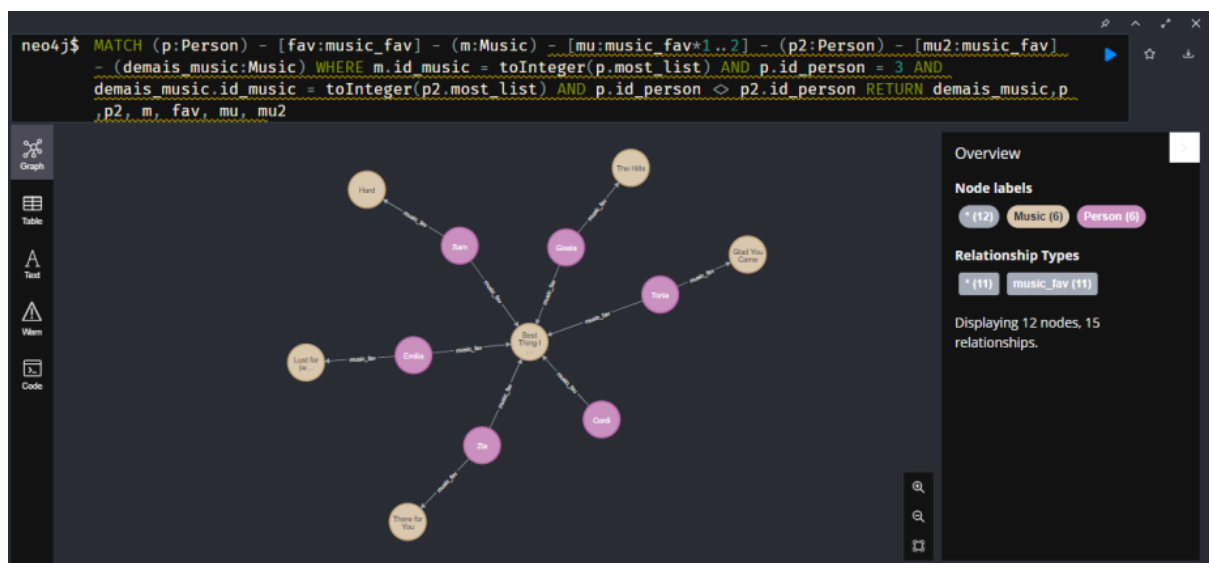


Imagem 27 : Teste no Neo4j da consulta 3

Verificamos também se a música principal da consulta, que no caso é a música favorita do usuário, está com o id correto, através das imagens 28 e 29 é possível perceber que os valores estão batendo.

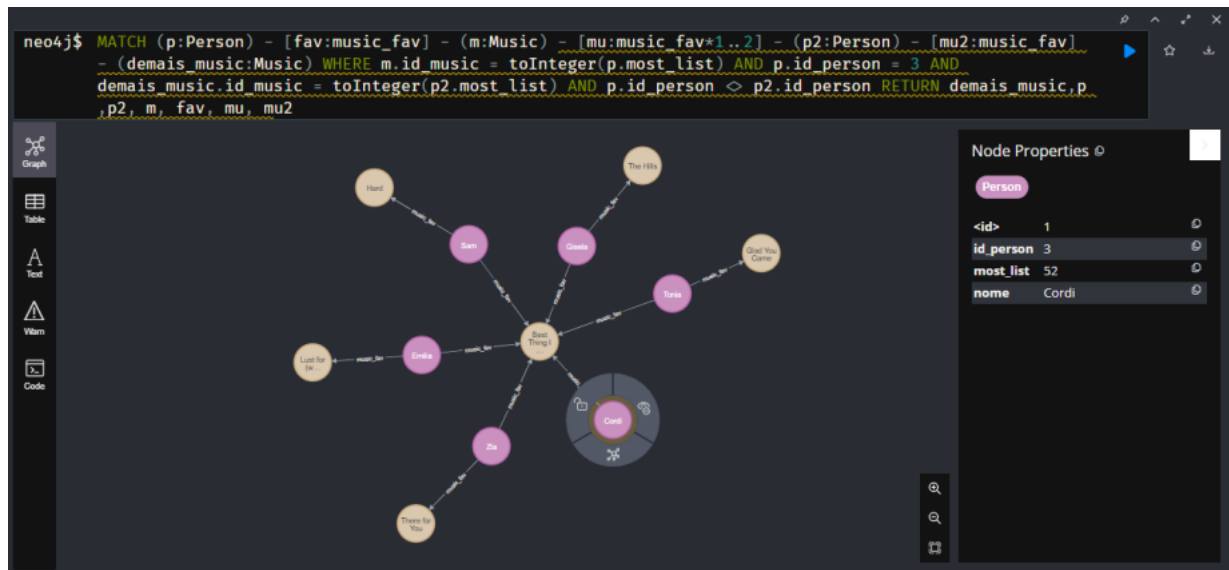


Imagem 28 : Teste no Neo4j para verificação do ID da música

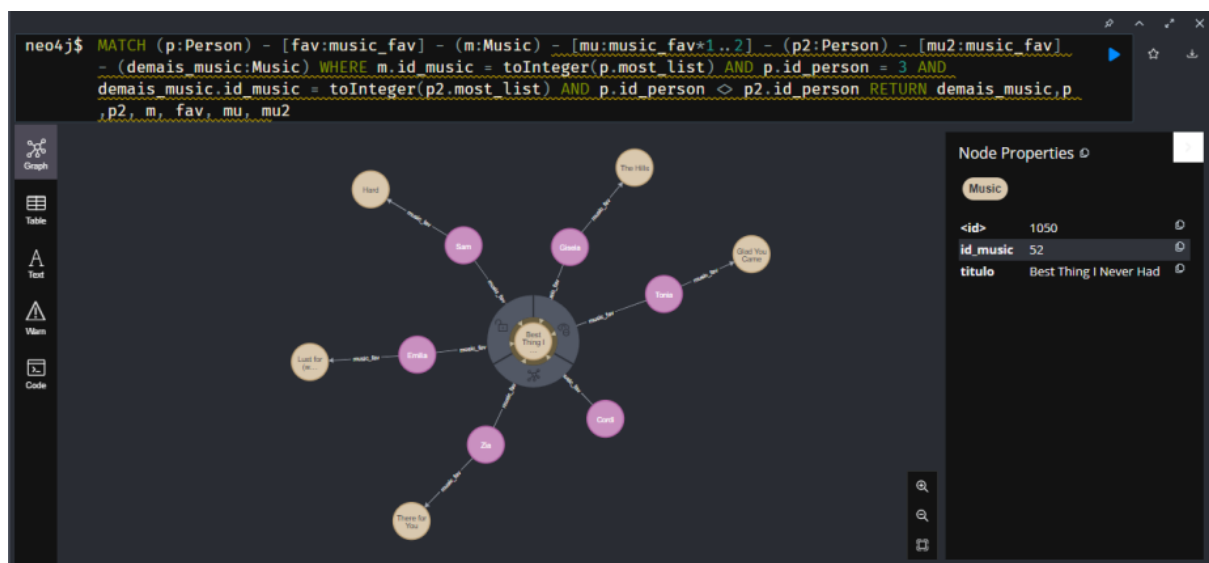


Imagem 29 : Teste no Neo4j para verificação do ID da música

Todas as informações esperadas foram retornadas pelo CouchDB (título, artista, gênero e ano), como é possível verificar na imagem 30:

```

Doesn't Mean Anything Alicia Keys hip pop 2010
Dog Days Are Over Florence + The Machine art pop 2010
End Game Taylor Swift pop 2018
I Wanna Go Britney Spears dance pop 2011
Impossible James Arthur pop 2013
Rise Katy Perry dance pop 2016
Sparks Hilary Duff dance pop 2015
Want To Dua Lipa dance pop 2018
  
```

Imagem 30 : Resultado da consulta 3

## 6.1.4 CONSULTA 4

Já na consulta quatro também é gerada uma playlist para o usuário, que é passado como parâmetro, mas esta é baseada nas músicas favoritas de seus amigos e dos amigos dos amigos em até três níveis de profundidade. Desta forma, o usuário consegue ter contato com músicas que já possa ter escutado, mas não tinham informações de nome ou artista, ou pode utilizar em uma festa, já que contará com músicas que são do agrado de seu núcleo de amigos.

```
#Consulta 4 - Retornar uma playlist levando em consideração as músicas favoritas de amigos em até três níveis de profundidade, o
idClientC4 = '865'
allIds = []

c4_mIds = spark.read\
    .format("org.neo4j.spark.DataSource")\
    .option("url", "neo4j://localhost:7687")\
    .option("authentication.type", "basic")\
    .option("authentication.basic.username", "neo")\
    .option("authentication.basic.password", "1234")\
    .option("query", "MATCH (p:Person) - [am:amigo_de*1..3] -> (p2:Person) - [mu:music_fav] -> (p2mu:Music) \
        WHERE p.id_person = '"+idClientC4+"' \
        RETURN p2mu.id_music")\
    .load()

for idmusic in c4_mIds.collect():
    allIds.append(str(idmusic[0]))

for doc in db.find({
    "selector": {
        "id_music": {
            "$or": allIds
        }
    },
    "limit": 200
}):
    print(doc["titulo"], doc["artista"], doc["genero"], doc["ano"])
```

Imagem 31 : Códigos da consulta 4

Para testar se o retorno da consulta no Neo4j estava correto utilizamos a mesma consulta exibindo todos os nós e relações e diminuimos a profundidade para 2 para uma melhor visualização.

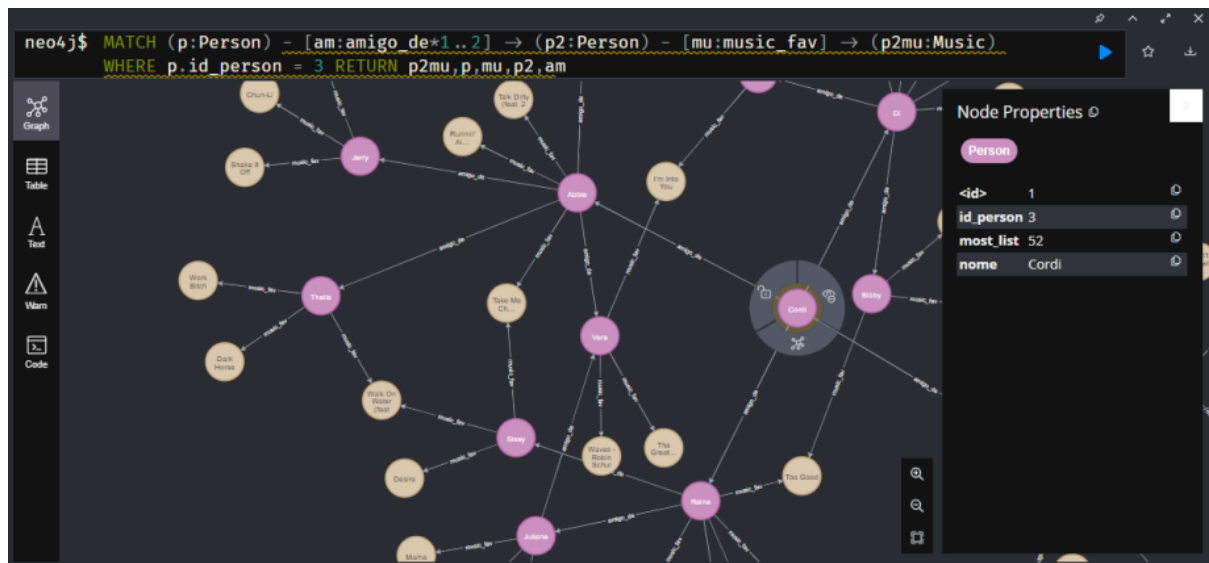


Imagem 32 : Teste no Neo4j para verificação da profundidade percorrida

Todas as informações esperadas foram retornadas pelo CouchDB (título, artista, gênero e ano), como é possível verificar na imagem 33:

```
All We Know The Chainsmokers electropop 2016
All of Me John Legend neo mellow 2014
Angel Fifth Harmony dance pop 2017
Anything Could Happen Ellie Goulding dance pop 2013
Atlas - From 掙he Hunger Games: Catching Fire?Soundtrack Coldplay permanent wave 2013
Attention Charlie Puth dance pop 2018
BURNITUP Janet Jackson dance pop 2016
Bad Liar Selena Gomez dance pop 2018
Bad Romance Lady Gaga dance pop 2010
Bang Bang Jessie J australian pop 2015
Beauty And A Beat Justin Bieber canadian pop 2012
Beneath Your Beautiful Labrinth pop 2013
Best Thing I Never Had Beyonce dance pop 2011
Blow Me (One Last Kiss) Pink dance pop 2012
Bodak Yellow Cardi B pop 2017
Body Moves DNCE dance pop 2017
Body Say Demi Lovato dance pop 2016
Bon appétit Katy Perry dance pop 2017
Boom Boom RedOne moroccan pop 2018
Break Free Ariana Grande dance pop 2015
```

Imagem 33 : Resultado da consulta 4

## 7. CONCLUSÃO

## 8. BIBLIOGRAFIA

- <https://www.psicoterapiaefins.com.br/2020/12/14/como-a-musica-pode-ajudar-a-lidar-com-as-emocoes/#:~:text=Ao%20escutarmos%20ou%20produzirmos%20m%C3%BAscas,associado%20ao%20aumento%20de%20imunidade>
- <https://gist.github.com/rioto9858/ff72b72b3bf5754d29dd1ebf898fc893>

- <https://extendsclass.com/csv-generator.html>