cesae digital

Centro para o Desenvolvimento de Competências Digitais

# Projeto de Base de dados


Formando:

Curso: Data Analyst

Data: 22/03/2024

# Índice

# Introdução

O presente projeto foi criado no âmbito da Unidade de Formação de Curta Duração (UFCD) de Gestão e Armazenamento de Dados. O propósito deste consistiu na criação de uma base de dados capaz de suportar funcionalidades analíticas e de gestão dos dados nela contidos.

Este projeto foi desenvolvido com recurso ao MySQL, um sistema de gestão de bases de dados que utiliza a linguagem SQL para criar, modelar e administrar a base de dados. Para a concretização deste trabalho, foi adotado o MySQL Workbench como a ferramenta principal de desenvolvimento e gestão do sistema.

Algumas das características do MySQL são o facto de ser um sistema compatível com a maioria dos sistemas operativos existentes, a sua velocidade na execução e armazenamento, bem como a sua portabilidade. Esta última, por sua vez, revela-se como uma vantagem significativa, uma vez que facilita a sua utilização em diferentes sistemas, plataformas e compiladores.

## Descrição do projeto

O projeto que decidi desenvolver consistiu no desenvolvimento de uma base de dados que se assemelha a uma plataforma de streaming de música. O objetivo principal foi criar um sistema capaz de armazenar informações detalhadas sobre os utilizadores e os artistas, proporcionando-lhes uma experiência personalizada e enriquecedora. Na base de dados, foram incluídos tabelas para registar os históricos de reprodução dos utilizadores, os eventos em que os artistas irão fazer, a discografia dos mesmos, possíveis playlists criadas pelos utilizadores e os géneros musicais a que pertencem as músicas.

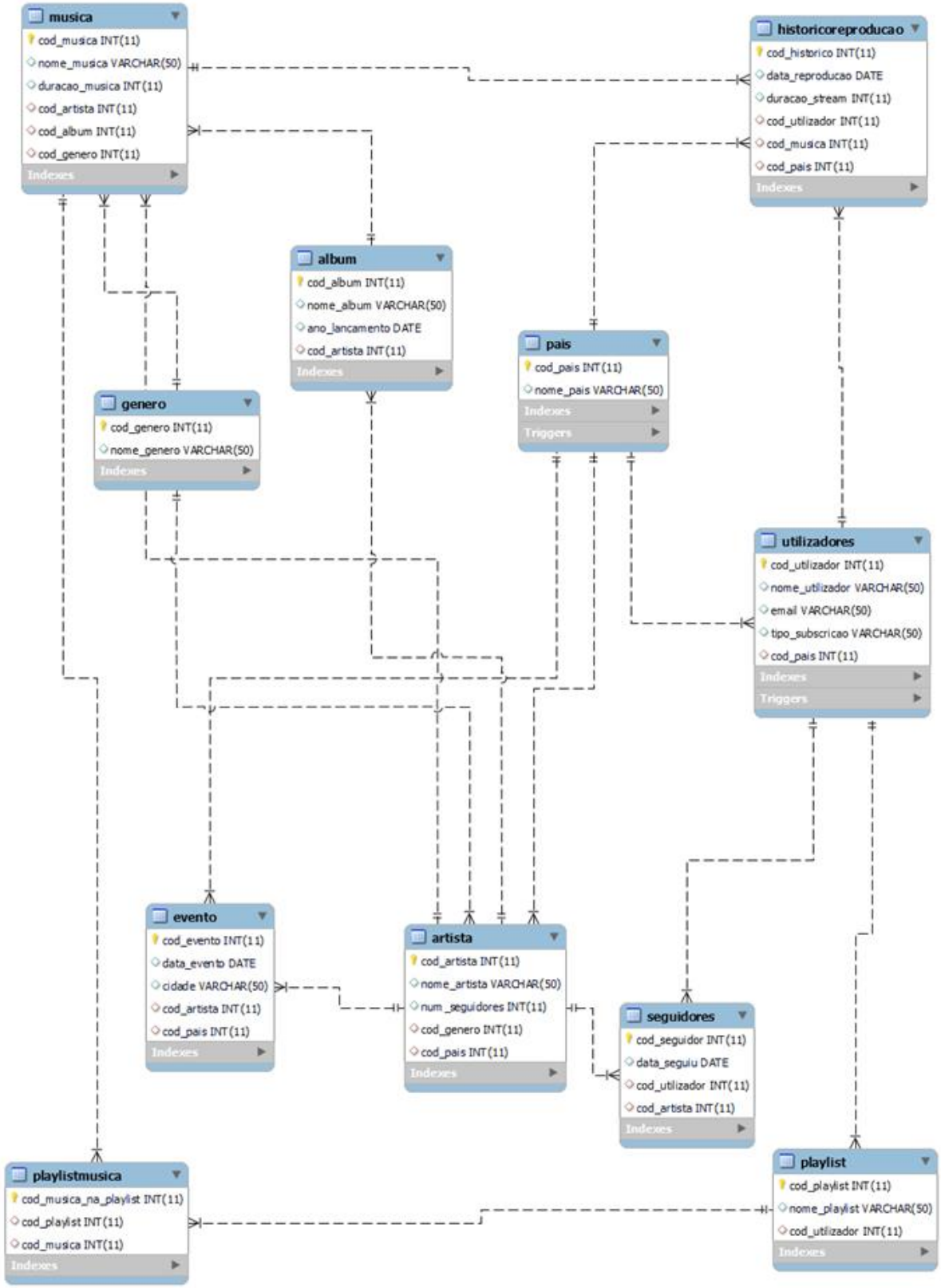
Na conceção desta base de dados, foram considerados vários elementos essenciais para garantir uma organização eficiente e abrangente dos dados. Para os artistas, foram incluídos campos para registar detalhes como a sua discografia, eventos nos quais participarão e possíveis associações a géneros musicais específicos. Por outro lado, para as músicas, foram contemplados dados como título, álbum ao qual pertencem e género musical.

Para enriquecer e diversificar os dados armazenados na minha base de dados, optei por utilizar diferentes ferramentas e linguagens. Inicialmente, explorei a API do Spotify for Developers para obter informações detalhadas sobre os artistas e as suas discografias. Usei linguagens como Python, para conseguir extrair dados como: nomes de músicas, álbuns, duração das faixas e até mesmo os géneros musicais associados aos artistas.

Além disso, para complementar as informações sobre eventos ao vivo, recorri ao Power BI para criar tabelas a partir de fontes online. Isso permitiu obter dados como datas e localizações de eventos, acrescentando uma dimensão adicional à minha base de dados.

Com uma fonte de dados mais abrangente e diversificada em mãos, o próximo passo foi organizar essas informações de forma estruturada e integrada à minha base de dados. Para isso, criei tabelas no MySQL Workbench, refletindo a estrutura dos dados obtidos. Uma vez definidas as tabelas, importei os dados diretamente a partir de um ficheiro CSV, gerado a partir do Excel.

# Modelo Relacional da Base de Dados



## Código do projeto

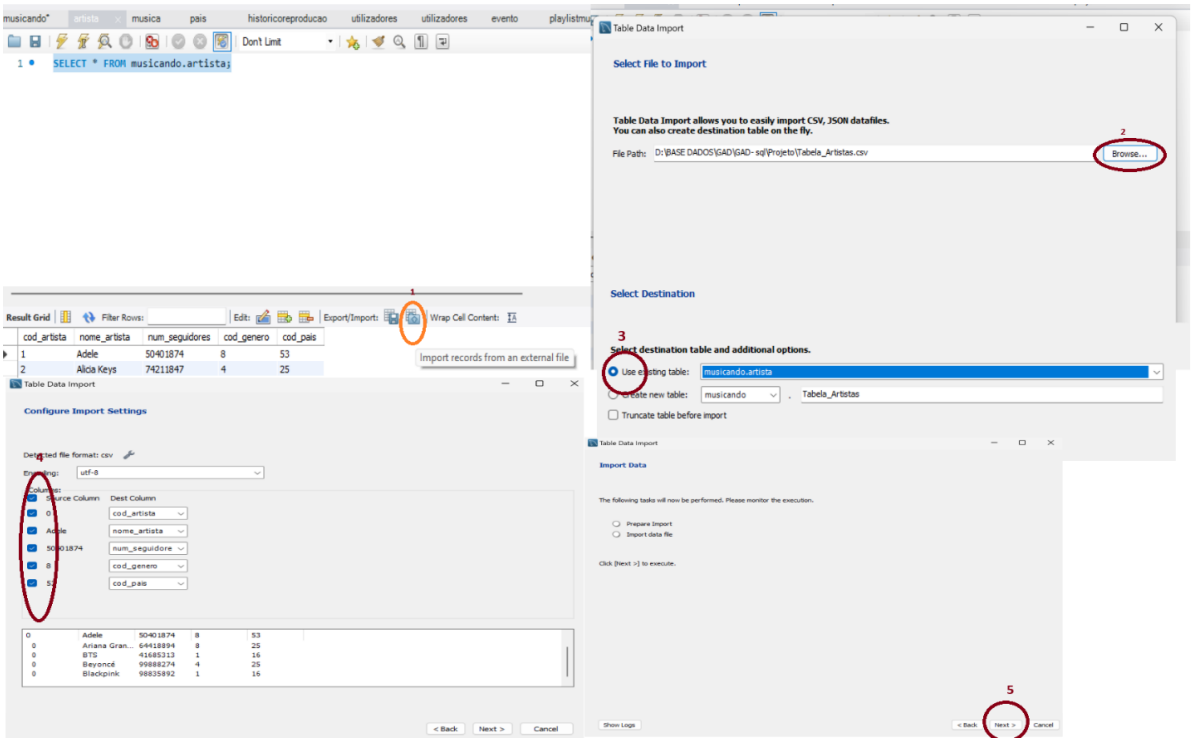
```
-- Para desactivar
SET SQL_SAFE_UPDATES = 0;

DROP DATABASE IF EXISTS Musicando;
CREATE DATABASE Musicando;
USE Musicando;

DROP TABLE IF EXISTS pais;
CREATE TABLE pais (
   cod_pais INT AUTO_INCREMENT PRIMARY KEY,
   nome_pais VARCHAR(50)
   );

DROP TABLE IF EXISTS genero;
CREATE TABLE genero (
   cod_genero INT AUTO_INCREMENT PRIMARY KEY,
   nome_genero VARCHAR(50)
   );

DROP TABLE IF EXISTS utilizadores;
CREATE TABLE utilizadores (
cod_utilizador INT AUTO_INCREMENT PRIMARY KEY,
nome_utilizador VARCHAR(50),
 email VARCHAR(50),
 tipo_subscricao VARCHAR(50),
 cod_pais int,
          FOREIGN KEY (cod_pais) REFERENCES pais(cod_pais));

DROP TABLE IF EXISTS artista;
CREATE TABLE artista (
   cod_artista INT AUTO_INCREMENT PRIMARY KEY,
   nome_artista VARCHAR(50),
   num_seguidores int,
   cod_genero int,
   cod_pais int,
   FOREIGN KEY (cod_genero) REFERENCES genero(cod_genero),
   FOREIGN KEY (cod_pais) REFERENCES pais(cod_pais));


DROP TABLE IF EXISTS evento;
CREATE TABLE evento (
   cod_evento INT AUTO_INCREMENT PRIMARY KEY,
   data_evento date,
   cidade varchar(50),
   cod_artista int,
   cod_pais int,
   FOREIGN KEY (cod_artista) REFERENCES artista(cod_artista),
   FOREIGN KEY (cod_pais) REFERENCES pais(cod_pais));
```

```
DROP TABLE IF EXISTS album;
CREATE TABLE album (
   cod_album INT AUTO_INCREMENT PRIMARY KEY,
   nome_album VARCHAR(50),
   ano_lancamento date,
   cod_artista int,
   FOREIGN KEY (cod_artista) REFERENCES artista(cod_artista));

DROP TABLE IF EXISTS musica;
CREATE TABLE musica (
   cod_musica INT AUTO_INCREMENT PRIMARY KEY,
   nome_musica VARCHAR(50),
   duracao_musica int,
   cod_artista int,
   cod_album int,
   cod_genero int,
   FOREIGN KEY (cod_artista) REFERENCES artista(cod_artista),
   FOREIGN KEY (cod_album) REFERENCES album(cod_album),
   FOREIGN KEY (cod_genero) REFERENCES genero(cod_genero));

DROP TABLE IF EXISTS playlist;
CREATE TABLE playlist (
   cod_playlist INT AUTO_INCREMENT PRIMARY KEY,
   nome_playlist VARCHAR(50),
   cod_utilizador int,
         FOREIGN KEY (cod_utilizador) REFERENCES utilizadores(cod_utilizador));

DROP TABLE IF EXISTS playlistMusica;
CREATE TABLE playlistMusica (
   cod_musica_na_playlist INT AUTO_INCREMENT PRIMARY KEY,
   cod_playlist int,
   cod_musica int,
   FOREIGN KEY (cod_playlist) REFERENCES playlist (cod_playlist),
   FOREIGN KEY (cod_musica) REFERENCES musica (cod_musica));


DROP TABLE IF EXISTS HistoricoReproducao;
CREATE TABLE HistoricoReproducao (
   cod_historico INT AUTO_INCREMENT PRIMARY KEY,
   data_reproducao date,
   duracao_stream int,
   cod_utilizador int,
   cod_musica int,
   cod_pais int,
   FOREIGN KEY (cod_utilizador) REFERENCES utilizadores(cod_utilizador),
   FOREIGN KEY (cod_musica) REFERENCES musica(cod_musica),
   FOREIGN KEY (cod_pais) REFERENCES pais(cod_pais));



DROP TABLE IF EXISTS seguidores;
```

```
CREATE TABLE seguidores (
   cod_seguidor INT AUTO_INCREMENT PRIMARY KEY,
   data_seguiu date,
   cod_utilizador int,
   cod_artista int,
   FOREIGN KEY (cod_utilizador) REFERENCES utilizadores(cod_utilizador),
   FOREIGN KEY (cod_artista) REFERENCES artista(cod_artista));

-- Para voltar a activar:
SET SQL_SAFE_UPDATES = 1;
```

## Tabelas de dados

Os dados foram inseridos na tabela da seguinte forma:




Apresento ainda aqui duas alternativas para inserir os dados, sendo que a primeira alternativa não tem tanto controle sobre o encoding do ficheiro, e a segunda são uma pequena amostra em relação à quantidade de dados criada para este projeto e presentes nos ficheiros csv.

```
-- Importar diretamente do ficheiro csv
-- LOAD DATA INFILE 'C:/Tabela_Paises.csv'
-- INTO TABLE pais
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO pais VALUES
(0, 'Uruguai'),
(0, 'Ucrânia'),
(0, 'Emirados Árabes Unidos'),
(0, 'Árabia Saudita'),
(0, 'Qatar'),
(0, 'Kuwait'),
(0, 'Omã'),
(0, 'Irão'),
(0, 'Iraque'),
(0, 'Afeganistão');

-- Importar diretamente do ficheiro csv
-- LOAD DATA INFILE 'C:/Tabela_Genero.csv'
```

```
-- INTO TABLE genero
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO genero VALUES
(0, 'Funk'),
(0, 'Gospel'),
(0, 'EDM'),
(0, 'Ambient'),
(0, 'Disco'),
(0, 'Techno'),
(0, 'House'),
(0, 'Grunge'),
(0, 'Dubstep'),
(0, 'World');

-- Importar diretamente do ficheiro csv
-- LOAD DATA INFILE 'C:/Tabela_Utilizadores.csv'
-- INTO TABLE utilizadores
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO utilizadores VALUES
(0,'MatildeRT','matildeRT@gmail.com','Standard',2),
(0, 'SkaterBoy8', 'skaterboy8@gmail.com', 'Standard', '3'),
(0, 'LuisaMusica100', 'LuisaMusica100@hotmail.com', 'Premium', '4'),
(0, 'MiguelFGR', 'MiguelFGR@hotmail.com', 'Standard', '5'),
(0, 'Beemusic', 'beemusic123@gmail.com', 'Premium', '1'),
(0, 'calebPT', 'calebpt@hotmail.com', 'Standard', '6'),
(0, 'anasoundw', 'anasoundw@gmail.com', 'Premium', '7'),
(0, 'Jackson2024', 'jackson2024@hotmail.com', 'Premium', '10'),
(0, 'HarperMain', 'harpersoul@gmail.com', 'Standard', '9'),
(0, 'antoniobanderas', 'antoniozorro@hotmail.com', 'Premium', '8');

-- Importar diretamente do ficheiro csv
-- LOAD DATA INFILE 'C:/Tabela_Artistas.csv'
-- INTO TABLE artista
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
insert into artista values
(0, 'Daniel Fernando', 50401874, 8, 3),
(0, 'Amália Rodrigues', 74211847, 4, 2),
(0, 'Anjos', 64418894, 1, 5),
(0, 'Quim Barreiros', 41685313, 2, 6),
(0, 'DZRT', 99888274, 10, 1),
(0, 'Ágata', 98835892, 9, 8),
(0, 'Xutos&Pontapés', 14803376, 3, 10),
(0, 'Salvador Sobral', 19074950, 7, 7),
```

```
(0, 'Toranja', 51048776, 5, 4),
(0, 'Doce', 58368137, 6, 9);

-- Importar diretamente do ficheiro csv
-- LOAD DATA INFILE "C:/Tabela_Eventos.csv"
-- INTO TABLE evento
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO evento VALUES
(0, '2024-04-08', 'New York City', 1,10),
(0, '2024-06-18', 'Florida', 2, 9),
(0, '2024-08-16', 'Inglewood', 3, 8),
(0, '2024-02-08', 'Nashville', 4, 7),
(0, '2024-03-06', 'New Orleans', 5, 6),
(0, '2024-05-02', 'Seattle', 6, 5),
(0, '2024-05-18', 'Tokyo', 7, 4),
(0, '2024-06-03', 'Orlando', 8, 3),
(0, '2024-03-19', 'Oklahoma City', 9, 2),
(0, '2024-10-31', 'Melbourne', 10, 1),
(0, '2024-02-07', 'Las Vegas', 1, 10);


-- Importar o ficheiro csv:
-- LOAD DATA INFILE "C:/Tabela_Album.csv"
-- INTO TABLE album
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO album VALUES
(0, "Mundo", '2001-10-18', 1),
(0, "DAlma", '2007-11-11', 2),
(0, "Amália Hoje", '2009-01-01', 3),
(0, "Aurora", '2004-04-19', 4),
(0, "Os Dias são a Noite", '2016-10-10', 5),
(0, "Rosa Negra", '1992-02-12', 6),
(0, "Sangue Oculto", '1997-02-14', 7),
(0, "Fado Promessa", '2008-12-01', 8),
(0, "Lisboa", '1996-06-25', 9),
(0, "Desfado", '2012-03-22', 10);


-- Importar diretamente do ficheiro csv:
-- LOAD DATA INFILE 'C:/Tabela_Musicas.csv'
-- INTO TABLE musica
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
```

```
INSERT INTO musica VALUES
(0, "Ai Se Ele Cai", 260, 1, 10, 1),
(0, "Meu Amor de Longe", 232, 2, 9, 2),
(0, "Desfado", 295, 3, 8, 3),
(0, "A Vida Toda", 187, 4, 7, 4),
(0, "Chuva", 241, 5, 6, 5),
(0, "Holy Grail", 338, 6, 5, 6),
(0, "E Depois do Adeus", 253,7, 4, 7),
(0, "O Homem do Leme", 83, 8, 3, 8),
(0, "Primavera", 167, 9, 2, 9),
(0, "Talking to the Moon", 217, 10, 1,10),
(0, "Amor a Portugal", 216, 1, 10, 1);

-- Importar diretamente do ficheiro csv:
-- LOAD DATA INFILE "C:/Tabela_Playlist.csv"
-- INTO TABLE playlist
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO playlist VALUES
(0, 'Saudade Sonora', 1),
(0, 'Noite Lisboeta', 2),
(0, 'Cores de Portugal', 3),
(0, 'Melodias Lusitanas', 10),
(0, 'Ondas do Fado', 9),
(0, 'Sabor a Mar', 8),
(0, 'Caminhos da Música Portuguesa', 4),
(0, 'Alma Lusa', 5),
(0, 'Jazz Jokes', 6),
(0, 'Vozes de Portugal', 7),
(0, 'Ritmos Tropicais', 1);

-- Importar diretamente do ficheiro csv:
-- LOAD DATA INFILE "C:/Tabela_MusicaNaPlaylist.csv"
-- INTO TABLE playlistMusica
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO playlistMusica VALUES
(0, 6, 5),
(0, 3, 8),
(0, 4, 4),
(0, 2, 1),
(0, 1, 9),
(0, 7, 7),
(0, 9, 2),
(0, 10, 6),
(0, 5, 3),
(0, 8, 10);
```

```
-- Importar diretamente do ficheiro csv:
-- LOAD DATA INFILE "C:/Tabela_Historico.csv"
-- INTO TABLE HistoricoReproducao
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO HistoricoReproducao VALUES
(0, '2024-11-09', 508, 1, 10, 8),
(0, '2024-12-22', 998, 2, 9, 10),
(0, '2025-02-14', 1837, 3, 8, 7),
(0, '2025-07-20', 1142, 4, 7, 1),
(0, '2025-07-02', 1146, 5, 6, 4),
(0, '2025-10-08', 168, 6, 5, 2),
(0, '2024-02-20', 122, 7, 4, 5),
(0, '2025-02-27', 207, 8, 3, 9),
(0, '2024-07-15', 967, 9, 2, 6),
(0, '2024-08-24', 327, 10, 1, 3);

-- Importar diretamente do ficheiro csv:
-- LOAD DATA INFILE "C:/ Tabela_Seguidores.csv"
-- INTO TABLE seguidores
-- FIELDS TERMINATED BY ';'
-- LINES TERMINATED BY '\n';

-- 10 novos valores que podem ser inseridos:
INSERT INTO seguidores VALUES
(0, '2024-11-09', 10, 8),
(0, '2024-12-22', 9, 10),
(0, '2025-02-14', 8, 7),
(0, '2025-07-20', 4, 7),
(0, '2025-07-02', 6, 4),
(0, '2025-10-08', 5, 2),
(0, '2024-02-20', 7, 5),
(0, '2025-02-27', 8, 3),
(0, '2024-07-15', 9, 6),
(0, '2024-08-24', 10, 3);
```

Passando agora para o trigger implementado, optei por criar um trigger para a tabela utilizador em que limitasse a inserção de um novo utilizador no caso de estar a ser inserido um nome de utilizador ou email já existente na base de dados:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `Evitar Duplicados` BEFORE INSERT ON `utilizadores`
FOR EACH ROW
BEGIN
   DECLARE email_existente INT;
   DECLARE utilizador_existente INT;
   -- Verifica se o email já está em uso
   SELECT COUNT(*) INTO email_existente
   FROM utilizadores
```

```
    WHERE email = NEW.email;
    -- Verifica se o nome de utilizador já está em uso
    SELECT COUNT(*) INTO utilizador_existente
    FROM utilizadores
    WHERE nome_utilizador = NEW.nome_utilizador;
    -- Se o email ou nome de utilizador já estiverem em uso, gera um erro
    IF email_existente > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Email já está em uso';
    END IF;
    IF utilizador_existente > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Nome de utilizador já está em uso';
    END IF;
END
```

Para o projeto em questão fez-se as seguintes funções:

```
-- 1) Total De Horas de Streaming
SELECT
    CONCAT(
        (SUM(duracao_stream) DIV 3600), ':',
        LPAD((SUM(duracao_stream) DIV 60) % 60, 2, '0'), ':',
        LPAD(SUM(duracao_stream) % 60, 2, '0')
    ) AS "Total De Horas de Streaming"
FROM historicoreproducao;

-- 2) Top 5 utilizadores que mais fazem stream
SELECT utilizadores.nome_utilizador as "Nome de Utilizador",
    CONCAT(
        (SUM(duracao_stream) DIV 3600), ':',
        LPAD((SUM(duracao_stream) DIV 60) % 60, 2, '0'), ':',
        LPAD(SUM(duracao_stream) % 60, 2, '0')
    ) AS "Horas por utilizador" FROM historicoreproducao
inner join utilizadores on historicoreproducao.cod_utilizador = utilizadores.cod_utilizador
GROUP BY historicoreproducao.cod_utilizador
ORDER BY SUM(duracao_stream) DESC
LIMIT 5;

-- 3) Que tipo de utilizador ouve mais Free ou Premium
SELECT utilizadores.tipo_subscricao as "Tipo de Subscrição",
    CONCAT(
        (SUM(duracao_stream) DIV 3600), ':',
        LPAD((SUM(duracao_stream) DIV 60) % 60, 2, '0'), ':',
        LPAD(SUM(duracao_stream) % 60, 2, '0')
    ) AS "Horas por tipo de utilizador" FROM historicoreproducao
inner join utilizadores on historicoreproducao.cod_utilizador = utilizadores.cod_utilizador
GROUP BY utilizadores.tipo_subscricao ORDER BY SUM(duracao_stream) DESC;
```

```
-- 4) Quais os gêneros mais ouvidos
SELECT genero.nome_genero as "Gênero",
   CONCAT(
      (SUM(historicoreproducao.duracao_stream) DIV 3600), ':',
      LPAD((SUM(historicoreproducao.duracao_stream) DIV 60) % 60, 2, '0'), ':',
      LPAD(SUM(historicoreproducao.duracao_stream) % 60, 2, '0')
   ) AS "Total de streaming por gênero"
FROM historicoreproducao
INNER JOIN musica ON historicoreproducao.cod_musica = musica.cod_musica
INNER JOIN artista ON musica.cod_artista = artista.cod_artista
INNER JOIN genero ON artista.cod_genero = genero.cod_genero
GROUP BY genero.cod_genero
ORDER BY SUM(historicoreproducao.duracao_stream) DESC;

-- 5) Em Portugal, em média, quanto tempo os utilizadores ouvem música?
SELECT pais.nome_pais as "Pais",
   CONCAT(
      (SUM(duracao_stream) / COUNT(duracao_stream)) DIV 3600, ':',
      LPAD(((SUM(duracao_stream) / COUNT(duracao_stream)) DIV 60) % 60, 2, '0'), ':',
      LPAD((SUM(duracao_stream) / COUNT(duracao_stream)) % 60, 2, '0')
   ) AS "Média de stream em Portugal" FROM historicoreproducao
INNER JOIN pais ON historicoreproducao.cod_pais = pais.cod_pais
WHERE pais.nome_pais = "Portugal";

-- 6) Quais os artistas mais ouvidos
SELECT artista.nome_artista as "Artista",
   CONCAT(
      (SUM(duracao_stream) DIV 3600), ':',
      LPAD((SUM(duracao_stream) DIV 60) % 60, 2, '0'), ':',
      LPAD(SUM(duracao_stream) % 60, 2, '0')
   ) AS "Artistas mais ouvidos" FROM historicoreproducao
INNER JOIN musica ON historicoreproducao.cod_musica = musica.cod_musica
inner join artista on musica.cod_artista = artista.cod_artista
GROUP BY artista.cod_artista
ORDER BY SUM(duracao_stream) DESC
LIMIT 5;

-- 7) Quais os artistas que lançaram album entre 2010 e 2015
select   artista.nome_artista as "Artista", album.nome_album as "Album", album.ano_lancamento
from album
inner join artista on album.cod_artista = artista.cod_artista
where year(ano_lancamento) between 2010 and 2015 order by ano_lancamento;

-- 8) Qual o artista com mais seguidores
```

```
SELECT MAX(num_seguidores) AS "Nº Seguidores", nome_artista AS "Artista" FROM artista  GROUP
BY nome_artista
ORDER BY MAX(num_seguidores) DESC limit 5 ;

-- 9) Qual os artistas, onde e quando que têm eventos entre Junho e Agosto
SELECT artista.nome_artista as "Artista", evento.data_evento as "Data", pais.nome_pais as "Pais",
evento.cidade as "Cidade" FROM evento
inner join artista on evento.cod_artista = artista.cod_artista
inner join pais on evento.cod_pais = pais.cod_pais
where month(data_evento) between 06 and 08 order by data_evento;

-- 10) Quais as música mais ouvida em Portugal em 2024
SELECT artista.nome_artista as "Artista", musica.nome_musica as "Música", pais.nome_pais as
"Pais",
   CONCAT(
      (SUM(duracao_stream) DIV 3600), ':',
      LPAD((SUM(duracao_stream) DIV 60) % 60, 2, '0'), ':',
      LPAD(SUM(duracao_stream) % 60, 2, '0')
   ) AS "Tempo Ouvido" FROM historicoreproducao
INNER JOIN musica ON historicoreproducao.cod_musica = musica.cod_musica
inner join artista on musica.cod_artista = artista.cod_artista
inner join pais on historicoreproducao.cod_pais = pais.cod_pais
WHERE pais.nome_pais = "Portugal" and year(historicoreproducao.data_reproducao)=2024
GROUP BY artista.cod_artista
ORDER BY SUM(duracao_stream) DESC
LIMIT 5;

-- 11) Qual o album da musica mais ouvida em Janeiro de 2025 na Alemanha
SELECT album.nome_album as "Album", musica.nome_musica as "Música", pais.nome_pais as
"Pais",
   CONCAT(
      (SUM(duracao_stream) DIV 3600), ':',
      LPAD((SUM(duracao_stream) DIV 60) % 60, 2, '0'), ':',
      LPAD(SUM(duracao_stream) % 60, 2, '0')
   ) AS "Tempo ouvido" FROM historicoreproducao
INNER JOIN musica ON historicoreproducao.cod_musica = musica.cod_musica
inner join artista on musica.cod_artista = artista.cod_artista
Inner join album on musica.cod_album = album.cod_album
inner join pais on historicoreproducao.cod_pais = pais.cod_pais
WHERE pais.nome_pais = "Alemanha" and year(historicoreproducao.data_reproducao)=2025 and
month(historicoreproducao.data_reproducao)=01
GROUP BY artista.cod_artista
ORDER BY SUM(duracao_stream) DESC
LIMIT 1;
```

```
-- 12) qual a playlist com mais músicas
SELECT playlist.nome_playlist AS "Playlist", COUNT(playlistmusica.cod_musica) AS "Número de
Músicas" FROM playlist
INNER JOIN playlistmusica ON playlist.cod_playlist = playlistmusica.cod_playlist
GROUP BY playlist.cod_playlist
ORDER BY COUNT(playlistmusica.cod_musica) DESC
LIMIT 1;

-- 13)  Quais os artistas com letra a começar por letra T
SELECT nome_artista as "Artista" FROM artista WHERE nome_artista LIKE 'T%';


-- 14) Qual a média de albums por artista
SELECT round(AVG(numero_de_albuns)) AS media_de_albuns_por_artista
FROM (
   SELECT artista.cod_artista, COUNT(album.cod_album) AS numero_de_albuns
   FROM artista
   INNER JOIN album ON artista.cod_artista = album.cod_artista
   GROUP BY artista.cod_artista
) AS subquery;


-- 15) Qual o artista que tem mais albuns
SELECT artista.nome_artista AS "Artista", COUNT(album.cod_album) AS "Número de Álbuns"
FROM artista
INNER JOIN album ON artista.cod_artista = album.cod_artista
GROUP BY artista.cod_artista
ORDER BY COUNT(album.cod_album) DESC
LIMIT 1;

-- 16) Qual o artista com menos albuns?
SELECT artista.nome_artista AS "Artista", COUNT(album.cod_album) AS "Número de Álbuns"
FROM artista
LEFT JOIN album ON artista.cod_artista = album.cod_artista
GROUP BY artista.cod_artista
ORDER BY COUNT(album.cod_album) ASC
LIMIT 1;

-- 17) Todas as músicas e respetivos albúns de um artista
SELECT artista.nome_artista as "Artista", musica.nome_musica AS "Música", album.nome_album
AS "Álbum"
FROM musica
JOIN album ON musica.cod_album = album.cod_album
JOIN artista ON album.cod_artista = artista.cod_artista
WHERE artista.nome_artista LIKE 'Taylor Swift';
```

```
-- 18) Mostra todos os seguidores de um artista
SELECT utilizadores.nome_utilizador AS "Nome do Utilizador", seguidores.data_seguiu as "Quando
Seguiu"
FROM seguidores
JOIN utilizadores ON seguidores.cod_utilizador = utilizadores.cod_utilizador
join artista on seguidores.cod_artista = artista.cod_artista
WHERE artista.nome_artista = 'Adele';

-- 19) Se quiser apagar um utilizador
DELETE FROM utilizadores WHERE cod_utilizador = 93;
```

## Conclusão

No final deste projeto, posso fazer um balanço positivo em relação aos objetivos a que me propus alcançar. Consegui realizar com sucesso as tarefas definidas, embora tenha enfrentado alguns desafios ao longo do caminho.

Durante o desenvolvimento do projeto, uma das principais dificuldades enfrentadas foi a gestão da grande quantidade de dados envolvidos, especialmente ao extrair informações do Spotify. Devido ao volume considerável de dados solicitados, houve ocasiões em que atingi o limite de pedidos permitidos pela API, resultando em um período de espera de 24 horas antes de poder fazer novas solicitações.

Para contornar esse obstáculo, foi necessário investir um esforço significativo na pesquisa de soluções eficientes e na implementação de estratégias que permitissem otimizar o processo de extração de dados. Isso incluiu a implementação de técnicas de limitação de solicitações e o desenvolvimento de mecanismos para armazenar e gerenciar os dados de forma eficaz.

Além disso, outra dificuldade que enfrentei foi a necessidade de lidar com métricas e conversões de dados em um contexto em que a escala dos números envolvidos era muito alta. Isso exigiu encontrar alternativas viáveis para evitar ultrapassar os limites para a formatação do tempo, uma vez que algumas das consultas pretendidas levariam a uma soma de horas superior à reconhecida pela formatação de tempo.

Apesar desses desafios, considero que o projeto foi concluído com sucesso e que os objetivos foram alcançados de forma satisfatória.

## Anexos/Código Extra

a) O seguinte código foi o utilizado para extrair informação do spotify

```
from dotenv import load_dotenv
import os
import base64
from requests import post, get
import json
from openpyxl import Workbook
import time  # Para adicionar um pequeno atraso entre as requisições

load_dotenv()

client_id = os.getenv("client_ID")
client_secret = os.getenv("client_secret")

def get_token():
    auth_string = client_id + ":" + client_secret
    auth_bytes = auth_string.encode("utf-8")
    auth_base64 = str(base64.b64encode(auth_bytes), "utf-8")

    url = "https://accounts.spotify.com/api/token"
    headers = {
        "Authorization": "Basic " + auth_base64,
        "Content-Type": "application/x-www-form-urlencoded"
    }
    data = {"grant_type": "client_credentials"}  # Fixed typo here
    result = post(url, headers=headers, data=data)
    json_result = json.loads(result.content)
    token = json_result["access_token"]  # Fixed typo here
    return token


def get_auth_header(token):
    return {"Authorization": "Bearer " + token}


def search_for_artist(token, artist_name):
    base_url = "https://api.spotify.com/v1/search"
    headers = get_auth_header(token)
    query = f"?q={artist_name}&type=artist&limit=1"

    query_url = base_url + query  # Corrected the URL construction
    result = get(query_url, headers=headers)
    json_result = json.loads(result.content)["artists"]["items"]

    if len(json_result) == 0:
        print(f"Não há artista com o nome '{artist_name}'")
        return None
    return json_result[0]

def get_albums_and_tracks_by_artist(token, artist_id):
    url =
f"https://api.spotify.com/v1/artists/{artist_id}/albums?limit=50&market=US
"
    headers = get_auth_header(token)
    result = get(url, headers=headers)

    # Verifica se a requisição foi bem-sucedida (status code 200)
```

```
    if result.status_code == 200:
        json_result = json.loads(result.content)

        all_albums_and_tracks = []
        for album in json_result['items']:
            album_info = {
                "album_name": album['name'],
                "release_date": album['release_date'],
                "type": album['album_type'],
                "tracks": get_tracks_from_album(token, album['id'])
            }
            all_albums_and_tracks.append(album_info)

        return all_albums_and_tracks
    else:
        print(f"Erro na requisição: {result.status_code}")
        # Espera 1 segundo antes de tentar novamente
        time.sleep(1)
        return None

def get_tracks_from_album(token, album_id):
    url = f"https://api.spotify.com/v1/albums/{album_id}/tracks"
    headers = get_auth_header(token)
    result = get(url, headers=headers)
    json_result = json.loads(result.content)
    return json_result.get('items', [])  # Ensure to return an empty list
if 'items' key is missing

def get_artist_genre(token, artist_id):
    url = f"https://api.spotify.com/v1/artists/{artist_id}"
    headers = get_auth_header(token)
    result = get(url, headers=headers)
    json_result = json.loads(result.content)
    return json_result.get('genres', [])  # Ensure to return an empty list
if 'genres' key is missing

def format_duration(ms):
    seconds = (ms / 1000) % 60
    minutes = (ms / (1000 * 60)) % 60
    return f"{int(minutes)}:{int(seconds):02d}"  # Format as MM:SS

def export_to_excel(artist_data):
    wb = Workbook()
    ws = wb.active
    ws.append(["Artist", "Genre", "Album", "Type", "Release Date",
"Track", "Duration"])

    for artist_name, albums_and_tracks in artist_data.items():
        if albums_and_tracks is None:
            continue  # Ignora se não houver álbuns e faixas

        for album_info in albums_and_tracks:
            artist_genre = ", ".join(get_artist_genre(token, artist_id))
            album_name = album_info['album_name']
            album_type = album_info['type']
            release_date = album_info['release_date']
            for track_info in album_info['tracks']:
                track_name = track_info['name']
                track_duration =
format_duration(track_info['duration_ms'])
                ws.append([artist_name, artist_genre, album_name,
```

```
album_type, release_date, track_name, track_duration])

    wb.save("spotify_data10.xlsx")

token = get_token()

# List of artists
#artists = ["Taylor Swift", "Adele", "Ed Sheeran"]
#artists = ["Beyoncé", "Shakira", "Justin Bieber"]
#artists = ["Bruno Mars", "Rihanna", "Eminem", "Celine Dion", "Luis
Fonsi", "Enrique Iglesias"]
#artists = ["Ariana Grande", "Jay-Z", "BTS", "Lady Gaga", "Maluma",
"Coldplay", "Drake", "Katy Perry"]
#artists = ["Alicia Keys", "U2", "Sia", "The Weeknd", "Mariah Carey",
"Juanes"]
#artists = ["Michael Bublé", "Madonna"]
#artists = ["Ricky Martin", "John Legend", "Pharrell Williams"]
#artists = ["Blackpink", "Elton John", "Olivia Rodrigo", "Ozuna", "Whitney
Houston"]
#artists = ["The Rolling Stones", "Daddy Yankee", "Bob Marley", "Queen"]
artists = [ "Red Hot Chili Peppers", "Bleachers", "Paramore"]

# Fetch data for each artist
artist_data = {}
for artist_name in artists:
    artist_info = search_for_artist(token, artist_name)
    if artist_info:
        artist_id = artist_info["id"]
        artist_data[artist_name] = get_albums_and_tracks_by_artist(token,
artist_id)

# Export data to Excel
export_to_excel(artist_data)
```

b) Tabela Álbum com uma amostra dos dados

| cod_album | nome_album | ano_lancamento | cod_artista |
|---|---|---|---|
| 1 | Empty Sky | 1969-06-06 | 15 |
| 2 | Elton John | 1970-04-10 | 15 |
| 3 | Get Yer Ya-Ya's Out! The Rolling Stones In Conc... | 1970-09-04 | 41 |
| 4 | Tumbleweed Connection | 1970-10-30 | 15 |
| 5 | Climb the Ladder | 1905-05-24 | 8 |
| 6 | 17/11/1970 | 1971-03-12 | 15 |
| 7 | Sticky Fingers (Super Deluxe) | 1971-04-23 | 41 |
| 8 | Sticky Fingers (Remastered) | 1971-04-23 | 41 |
| 9 | Madman Across The Water | 1971-11-05 | 15 |
| 10 | Jamming With Edward | 1972-01-07 | 41 |
| 11 | Exile On Main Street (2010 Re-Mastered) | 1972-05-12 | 41 |
| 12 | Exile On Main Street (Deluxe Version) | 1972-05-12 | 41 |
| 13 | Honky Chateau | 1972-05-19 | 15 |
| 14 | Honky Château (50th Anniversary Edition) | 1972-05-19 | 15 |
| 15 | Don't Shoot Me I'm Only The Piano Player | 1973-01-22 | 15 |

c) Tabela artista com uma amostra dos dados

| cod_artista | nome_artista | num_seguidores | cod_genero | cod_pais |
|---|---|---|---|---|
| 1 | Adele | 50401874 | 8 | 53 |
| 2 | Alicia Keys | 74211847 | 4 | 25 |
| 3 | Ariana Grande | 64418894 | 8 | 25 |
| 4 | BTS | 41685313 | 1 | 16 |
| 5 | Beyoncé | 99888274 | 4 | 25 |
| 6 | Blackpink | 98835892 | 1 | 16 |
| 7 | Bleachers | 14803376 | 3 | 25 |
| 8 | Bob Marley | 19074950 | 7 | 37 |
| 9 | Bruno Mars | 51048776 | 8 | 25 |
| 10 | Celine Dion | 58368137 | 8 | 11 |
| 11 | Coldplay | 58325750 | 8 | 53 |
| 12 | Daddy Yankee | 5185178 | 8 | 51 |
| 13 | Drake | 57067134 | 4 | 11 |
| 14 | Ed Sheeran | 113118555 | 8 | 53 |
| 15 | Elton John | 13787480 | 8 | 53 |

d) Tabela evento com uma amostra dos dados

| cod_evento | data_evento | cidade | cod_artista | cod_pais |
|---|---|---|---|---|
| 1 | 2024-02-07 | Tokyo | 40 | 38 |
| 2 | 2024-02-08 | Tokyo | 40 | 38 |
| 3 | 2024-02-09 | Tokyo | 40 | 38 |
| 4 | 2024-02-10 | Tokyo | 40 | 38 |
| 5 | 2024-02-16 | Melbourne | 40 | 4 |
| 6 | 2024-02-17 | Melbourne | 40 | 4 |
| 7 | 2024-02-18 | Melbourne | 40 | 4 |
| 8 | 2024-02-23 | Sydney | 40 | 4 |
| 9 | 2024-02-24 | Sydney | 40 | 4 |
| 10 | 2024-02-25 | Sydney | 40 | 4 |
| 11 | 2024-02-26 | Sydney | 40 | 4 |
| 12 | 2024-03-02 | Singapore | 40 | 58 |
| 13 | 2024-03-03 | Singapore | 40 | 58 |
| 14 | 2024-03-04 | Singapore | 40 | 58 |
| 15 | 2024-03-07 | Singapore | 40 | 58 |

e) Tabela genero com uma amostra dos dados

| cod_genero | nome_genero |
|---|---|
| 1 | K-pop |
| 2 | Rap |
| 3 | Rock |
| 4 | R&B |
| 5 | Latin Trap |
| 6 | Jazz |
| 7 | Reggae |
| 8 | Pop |
| 9 | Alternative Rock |
| 10 | Hip-Hop |
| 11 | Latin trap |
| 12 | Índie |
| 13 | Electronic |
| 14 | Folk |
| 15 | Country |

genero 1

f) Tabela histórico de reproduçao com uma amostra dos dados

| cod_historico | data_reproducao | duracao_stream | cod_utilizador | cod_musica | cod_pa |
|---|---|---|---|---|---|
| 1 | 2024-06-19 | 23619 | 46 | 11281 | 54 |
| 2 | 2025-02-08 | 16347 | 42 | 3402 | 22 |
| 3 | 2025-05-01 | 48693 | 27 | 10291 | 62 |
| 4 | 2025-08-29 | 46509 | 42 | 5723 | 22 |
| 5 | 2025-05-14 | 43785 | 71 | 7227 | 37 |
| 6 | 2025-01-10 | 44842 | 44 | 11795 | 11 |
| 7 | 2025-12-24 | 52107 | 35 | 11787 | 29 |
| 8 | 2025-10-18 | 2916 | 16 | 11437 | 20 |
| 9 | 2025-10-28 | 18831 | 7 | 988 | 13 |
| 10 | 2025-05-18 | 5104 | 19 | 4447 | 13 |
| 11 | 2025-04-11 | 23550 | 17 | 11364 | 61 |
| 12 | 2025-02-11 | 46920 | 65 | 4418 | 63 |
| 13 | 2025-12-07 | 31451 | 58 | 607 | 48 |
| 14 | 2024-07-25 | 18030 | 18 | 830 | 8 |
| 15 | 2024-11-02 | 34830 | 24 | 6487 | 10 |

coreproducao 1

g) Tabela musica com uma amostra dos dados

| cod_musica | nome_musica | duracao_musica | cod_artista | cod_album | cod_genero |
|---|---|---|---|---|---|
| 1 | "40" - Live From Sankt Goarsha... | 227 | 43 | 820 | 19 |
| 2 | "40" - Remastered 2008 | 157 | 43 | 77 | 15 |
| 3 | "40" - Remastered 2008 | 157 | 43 | 76 | 8 |
| 4 | "Hooray ! Hooray ! It's a Holi-H... | 189 | 36 | 136 | 16 |
| 5 | "Little Mimi's" Theme | 17 | 28 | 677 | 19 |
| 6 | "Slut!" (Taylor's Version) (From ... | 180 | 40 | 813 | 3 |
| 7 | "Slut!" (Taylor's Version) (From ... | 180 | 40 | 814 | 21 |
| 8 | #Beautiful | 199 | 28 | 499 | 21 |
| 9 | '39 - Live | 206 | 34 | 56 | 11 |
| 10 | '39 - Live At Earl's Court, Londo... | 226 | 34 | 35 | 19 |
| 11 | '39 - Remastered 2011 | 210 | 34 | 34 | 17 |
| 12 | '39 - Remastered 2011 | 210 | 34 | 406 | 19 |
| 13 | '39 - Remastered 2011 | 210 | 34 | 35 | 4 |
| 14 | '97 Bonnie & Clyde | 316 | 16 | 190 | 15 |
| 15 | '97 Bonnie & Clyde | 316 | 16 | 189 | 1 |

h) Tabela pais com uma amostra dos dados

| cod_pais | nome_pais |
|---|---|
| 1 | África do Sul |
| 2 | Alemanha |
| 3 | Argentina |
| 4 | Austrália |
| 5 | Áustria |
| 6 | Bahamas |
| 7 | Barbados |
| 8 | Brasil |
| 9 | Bulgaria |
| 10 | Bélgica |
| 11 | Canadá |
| 12 | Chile |
| 13 | China |
| 14 | Chipre |
| 15 | Colômbia |

pais 1

i) Tabela playlist com uma amostra dos dados

Result Grid | Filter Rows: | Edit:

| cod_playlist | nome_playlist | cod_utiliza |
|---|---|---|
| 1 | Rock & Roll with It | 1 |
| 2 | Pop and Lock | 18 |
| 3 | R&B - Rhythm and Banter | 26 |
| 4 | Beats and Banter | 30 |
| 5 | Samba and Smiles | 13 |
| 6 | Indie-Inspired Laughs | 7 |
| 7 | Funk Frenzy | 5 |
| 8 | Hip Hop Humor | 23 |
| 9 | Jazz Jokes | 7 |
| 10 | Country Comedy | 12 |
| 11 | Folk Fun | 17 |
| 12 | Reggae Riffs and Roasts | 27 |
| 13 | Electro-Eclectic Entertainment | 1 |
| 14 | Salsa and Chuckles | 22 |
| 15 | Disco Delights | 10 |

j) Tabela playslistmusica com uma amostra dos dados

| cod_musica_na_playlist | cod_playlist | cod_musica |
|---|---|---|
| 1 | 60 | 5992 |
| 2 | 32 | 8470 |
| 3 | 41 | 9904 |
| 4 | 28 | 11251 |
| 5 | 30 | 9295 |
| 6 | 57 | 7316 |
| 7 | 27 | 10206 |
| 8 | 47 | 6901 |
| 9 | 27 | 12386 |
| 10 | 53 | 10727 |
| 11 | 52 | 1685 |
| 12 | 16 | 10837 |
| 13 | 30 | 10290 |
| 14 | 36 | 6417 |
| 15 | 37 | 4182 |

k) Tabela seguidores com uma amostra dos dados

| cod_seguidor | data_seguiu | cod_utilizador | cod_artis |
|---|---|---|---|
| 1 | 2024-01-14 | 16 | 29 |
| 2 | 2022-09-17 | 19 | 43 |
| 3 | 2018-11-04 | 16 | 14 |
| 4 | 2018-04-22 | 28 | 39 |
| 5 | 2021-06-23 | 10 | 13 |
| 6 | 2021-06-27 | 3 | 17 |
| 7 | 2024-01-01 | 5 | 19 |
| 8 | 2020-11-20 | 3 | 35 |
| 9 | 2020-01-07 | 19 | 33 |
| 10 | 2020-02-20 | 10 | 19 |
| 11 | 2019-01-30 | 13 | 31 |
| 12 | 2023-07-08 | 13 | 33 |
| 13 | 2023-12-21 | 15 | 35 |
| 14 | 2020-06-29 | 25 | 43 |
| 15 | 2021-08-12 | 26 | 34 |

l) Tabela utilizadores com uma amostra dos dados

| cod_utilizador | nome_utilizador | email | tipo_subscricao | cod_pais |
|---|---|---|---|---|
| 1 | SophiaMovieManiac | melodyadventure22@gmail.com | Premium | 30 |
| 2 | DanielFashionista | johntechgeek@gmail.com | Standard | 2 |
| 3 | ChloeFoodieForever | sarahstargazer@hotmail.com | Premium | 57 |
| 4 | AidenDIYEnthusiast | dylancoffeeaddict@hotmail.com | Standard | 58 |
| 5 | IsabellaBeachBum | emilybookworm123@gmail.com | Premium | 57 |
| 6 | CalebPhotographyFan | liamfitnessfanatic@hotmail.com | Standard | 41 |
| 7 | EllaChessPlayer | zoenatureexplorer@gmail.com | Premium | 13 |
| 8 | JacksonCookingWizard | lucasgamerpro_x@hotmail.com | Premium | 19 |
| 9 | HarperCyclistLife | sophiaartisticsoul@gmail.com | Standard | 18 |
| 10 | MasonGreenThumb | ethansciencebuff@hotmail.com | Premium | 15 |
| 11 | AddisonSkyDiver | audreytravelenthusiast@gmail.com | Standard | 43 |
| 12 | LiamMysteryNovelist | maxmovie-maniac84@hotmail.com | Premium | 66 |
| 13 | ZoeySoccerStar | averyfashionista22@gmail.com | Premium | 56 |
| 14 | SamuelMountainClimber | oliverfoodieforever@hotmail.com | Standard | 64 |
| 15 | AvaHappyHiker | miaDIYenthusiast@gmail.com | Premium | 33 |

# Bibliografia/Siteografia

https://www.w3schools.com/sql/

https://downloads.mysql.com/docs/refman-4.1-pt.a4.pdf

https://developer.spotify.com/documentation/web-api

https://www.youtube.com/watch?v=WAmEZBEeNmg&ab_channel=AkamaiDeveloper

https://www.khanacademy.org/computing/computer-programming/sql