QUESTÃO 1

Estamos desenvolvendo um sistema de banco de dados abrangente para gerenciar informações relacionadas à indústria cinematográfica. Este sistema permitirá a catalogação de filmes, incluindo todos os detalhes necessários para aficionados por cinema, críticos, produtores e acadêmicos. O banco de dados projetado precisa armazenar informações sobre cada filme, como título, duração, ano de lançamento e uma sinopse breve que fornece um panorama do enredo.

Além disso, o sistema deve manter registros detalhados dos diretores, produtores e atores de cada filme, incluindo seus nomes, datas de nascimento e nacionalidades, além disso para produtores em específico existe um nome de empresa relacionado. Note que um filme tem no minimo um produtor e um diretor e pode ter de nenhum a diversos atores envolvidos.

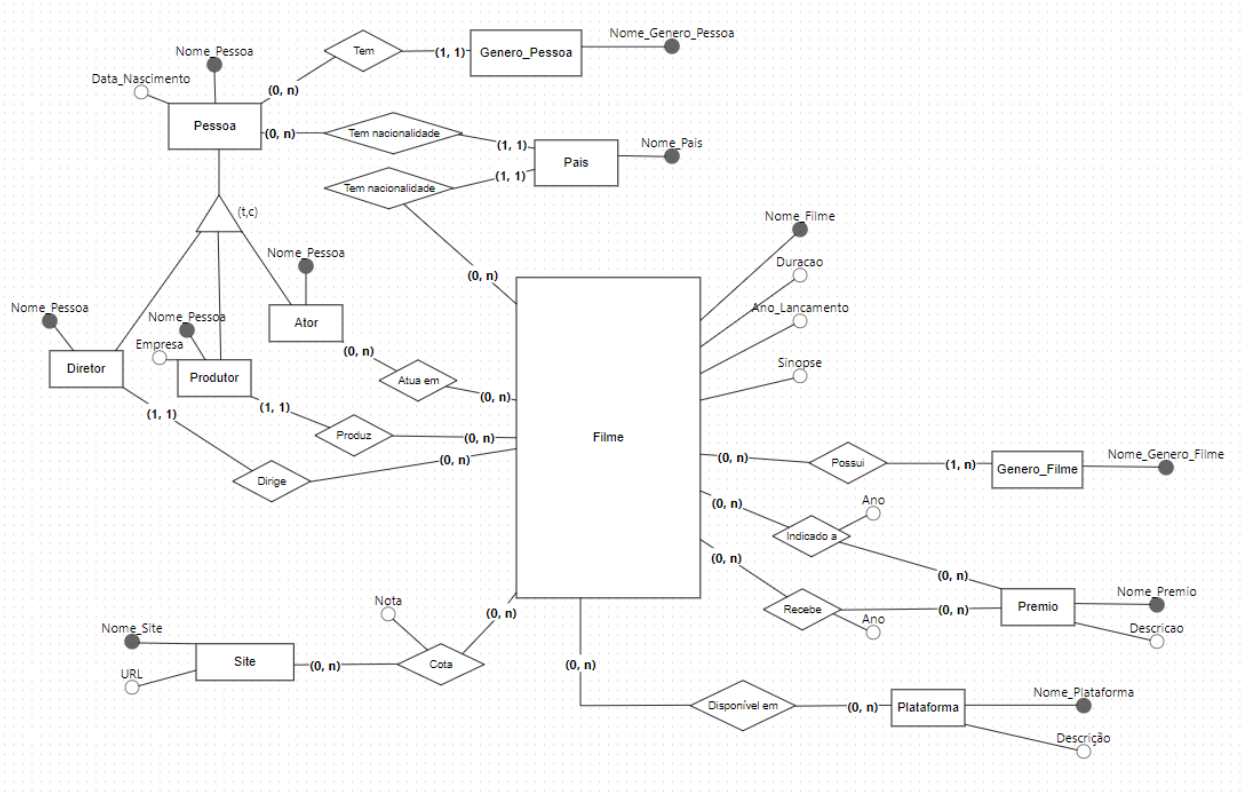
O sistema também categoriza os filmes por gênero (comédia, terror, etc.), facilitando buscas específicas para usuários interessados em particularidades, note que um filme pode ter mais de um gênero, mas apenas um tipo. Cada filme também será associado ao país onde foi feito.

Para facilitar o acesso ao conteúdo, o banco de dados incluirá informações sobre as plataformas em que cada filme está disponível, como Netflix, Amazon Prime ou cinemas locais, além de conter uma descrição sobre a plataforma em questão.

Os filmes no banco de dados serão associados não só aos prêmios que receberam, mas também às indicações que obtiveram, com descrições detalhadas dos prêmios, os anos de recebimento e os anos de indicação. Sites de cotação de filmes, como IMDB ou Rotten Tomatoes, serão conectados ao banco, permitindo que os usuários vejam avaliações agregadas e específicas de diferentes fontes.

Este sistema de banco de dados será uma ferramenta indispensável para quem procura informações completas sobre filmes, oferecendo uma maneira integrada de explorar a vasta paisagem cinematográfica através de múltiplos ângulos e perspectivas.

QUESTÃO 2



Link:https://app.brmodeloweb.com/#!/publicview/6643e4daad46d50b6b47264e

O banco de dados será composto pelas seguintes entidades e relacionamentos:

Tabelas

Filme

Titulo (PK): Título do filme. // Alfanumérico
Duracao: Duração do filme (em minutos). // Numérico (inteiro em minutos)
Ano_Lancamento: Ano de lançamento do filme. // Numérico (ano)

Sinopse: Descrição breve do filme. // Alfanumérico (texto)

Pessoa

Nome_Pessoa (PK): Nome da pessoa. // Alfabético
Data_Nascimento: Data de nascimento da pessoa. // Data (dia-mês-ano)
Nacionalidade: Nacionalidade da pessoa. // Alfabético

Diretor

Nome_Pessoa (PK, FK): Nome da pessoa, que é um diretor. // Alfabético

Produtor

Nome_Pessoa (PK, FK): Nome da pessoa, que é um produtor. // Alfabético
Empresa: Empresa associada ao produtor. // Alfanumérico

Ator

Nome_Pessoa (PK, FK): Nome da pessoa, que é um ator. // Alfabético

Genero_Filme

Nome_Genero_Filme (PK): Nome do gênero (filme). // Alfabético

Genero_Pessoa

Nome_Genero_Filme (PK): Nome do gênero (pessoa). // Alfabético

Pais

Nome_Pais (PK): Nome do país. // Alfabético

Plataforma

Nome_Plataforma (PK): Nome da plataforma (Netflix, Amazon, YouTube, etc.). // Alfanumérico
Descricao: Descrição/Info sobre a plataforma (Ex: Plataforma de streaming criada em 2006). // Alfabético

Site

Nome_Site (PK): Nome do site de cotação (e.g., IMDB, Rotten Tomatoes). // Alfabético
URL: URL do site de cotação. // Alfanumérico

Premio

Nome_Premio (PK): Nome do prêmio. // Alfanumérico
Descricao: Descrição do prêmio. // Alfanumérico

Relacionamentos

Filme_Genero

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Genero (FK): Referência ao gênero. // Alfabético

Filme_Pais

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Pais (FK): Referência ao país do filme. // Alfabético

Filme_Produtor

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Produtor (FK): Referência ao produtor. // Alfabético

Filme_Diretor

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Diretor (FK): Referência ao diretor. // Alfabético

Filme_Ator

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Ator (FK): Referência ao ator. // Alfabético

Filme_Plataforma

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Plataforma (FK): Referência à plataforma. // Alfanumérico

Filme_Site

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Site (FK): Referência ao site de cotação. // Alfabético
Nota: Nota dada ao filme no site de cotação. // Numérico (0 a 100)

Filme_Indicacao

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Premio (FK): Referência ao prêmio. // Alfanumérico
Ano: Ano da indicação. // Numérico (4 dígitos inteiros), ano

Filme_Premio

Titulo_Filme (FK): Referência ao filme. // Alfanumérico
Nome_Premio (FK): Referência ao prêmio. // Alfanumérico
Ano: Ano em que o prêmio foi recebido. // Numérico (4 dígitos inteiros), ano

Pessoa_Pais

Nome_Pessoa (FK): Referência à pessoa. // Alfabético

Nome_Pais (FK): Referência ao país da pessoa. // Alfabético

QUESTÃO 3

1. Um Filme Não Pode Ser Lançado Antes do Nascimento do Diretor

Regra: Ao inserir ou atualizar um filme, verificar se a data de lançamento do filme é posterior à data de nascimento do diretor associado.

Justificativa: Esta regra assegura que as datas relacionadas ao filme e ao diretor são coerentes e realistas.

2. Um Ator Não Pode Participar de Filmes Lançados Antes do Seu Nascimento

Regra: Ao inserir ou atualizar uma participação de ator em um filme, verificar se a data de nascimento do ator é anterior à data de lançamento do filme.

Justificativa: Esta regra assegura que as datas relacionadas aos filmes e aos atores são coerentes e realistas.

3. Validação de Duração do Filme

Regra: A duração de um filme deve ser maior que 0 e menor que 300 minutos.

Justificativa: Filmes normalmente têm uma duração que varia de poucos minutos (curtas) a algumas horas. Estabelecer uma duração mínima e máxima evita erros de entrada de dados, como a inserção de valores inválidos (e.g., 0 ou um número excessivamente alto).

4.Limite de Prêmios Recebidos

Regra:
Um filme não pode receber prêmios em anos muito diferentes.

Justificativa:
Essa regra garante que a atribuição de prêmios seja coerente com a relevância temporal do filme. Prêmios distribuídos em anos muito distantes podem indicar inconsistências na avaliação de um filme, ou a falta de atualizações adequadas nos critérios de premiação. Estabelecer um limite de tempo evita esses problemas, assegurando que os prêmios reflitam um período de avaliação.

5.Ano de Lançamento do Filme

Regra: Nenhum filme pode ter sido lançado antes de 1888 e o ano de lançamento deve ser um valor de quatro dígitos.

Justificativa: Esta regra assegura que as datas relacionadas ao filme são coerentes e realistas, uma vez que a primeira exibição de um filme ocorreu em 1888 e a data de lançamento deve ser um valor válido de quatro dígitos.

QUESTÃO 4

Obs: Sublinhado = PK , Em **negrito** = **FK**. (fk) (pk)

FILME (Titulo, Duracao, Ano_Lancamento, Sinopse, **Nome_Pais**)

PESSOA (Nome_Pessoa, Data_Nascimento, **Nome_Pais**, **Nome_Genero_Pessoa**)

DIRETOR (**Nome_Pessoa**)
PRODUTOR (**Nome_Pessoa**, Empresa)
ATOR (**Nome_Pessoa**)

GENERO_FILME (Nome_Genero_Filme)

GENERO_PESSOA (Nome_Genero_Filme)

PAIS (Nome_Pais)

PLATAFORMA (Nome_Plataforma, Descricao)

SITE (Nome_Site, URL)

PREMIO (Nome_Premio, Descricao)

COTACAO (**Titulo_Filme**, **Nome_Site**, Nota)

INDICACAO (**Titulo_Filme**, **Nome_Premio**, **Ano**)

PREMIACAO (**Titulo_Filme**, **Nome_Premio**, **Ano**)

FILME_PLATAFORMA (**Titulo_Filme**, **Nome_Plataforma**)

FILME_PRODUTOR (**Titulo_Filme**, **Nome_Pessoa**)

FILME_DIRETOR (**Titulo_Filme**, **Nome_Pessoa**)

FILME_ATOR (**Titulo_Filme**, **Nome_Pessoa**)

FILME_GENERO (**Titulo_Filme**, **Nome_Genero_Filme**)

QUESTÃO 5/6

CRIAÇÃO COMPLETA COM ALTER TABLE:

```
CREATE SCHEMA filmes;
```

```sql
CREATE TABLE filmes.Pais (
    Nome_Pais VARCHAR(255) NOT NULL
);

CREATE TABLE filmes.Genero_Pessoa (
    Nome_Genero_Pessoa VARCHAR(255) NOT NULL
);

CREATE TABLE filmes.Genero_Filme (
    Nome_Genero_Filme VARCHAR(255) NOT NULL
);

CREATE TABLE filmes.Pessoa (
    Nome_Pessoa VARCHAR(255) NOT NULL UNIQUE,
    Data_Nascimento DATE,
    Nome_Pais VARCHAR(255),
    Nome_Genero_Pessoa VARCHAR(255)
);

CREATE TABLE filmes.Diretor (
    Nome_Pessoa VARCHAR(255) NOT NULL UNIQUE
);

CREATE TABLE filmes.Produtor (
    Nome_Pessoa VARCHAR(255) NOT NULL UNIQUE,
    Empresa VARCHAR(255)
);

CREATE TABLE filmes.Ator (
    Nome_Pessoa VARCHAR(255) NOT NULL UNIQUE
);

CREATE TABLE filmes.Plataforma (
    Nome_Plataforma VARCHAR(255) NOT NULL,
    Descricao TEXT
);

CREATE TABLE filmes.Site (
    Nome_Site VARCHAR(255) NOT NULL,
    URL VARCHAR(255)
```

```
);

CREATE TABLE filmes.Premio (
    Nome_Premio VARCHAR(255) NOT NULL,
    Descricao TEXT
);

CREATE TABLE filmes.Filme (
    Titulo VARCHAR(255) NOT NULL,
    Duracao INT CHECK (Duracao > 0 AND Duracao < 300),
    Ano_Lancamento INT CHECK (Ano_Lancamento > 1888 AND Ano_Lancamento
BETWEEN 1000 AND 9999),
    Sinopse TEXT,
    Nome_Pais VARCHAR(255)
);

ALTER TABLE filmes.Pais ADD CONSTRAINT PK_Pais PRIMARY KEY (Nome_Pais);
ALTER TABLE filmes.Genero_Pessoa ADD CONSTRAINT PK_Genero_Pessoa PRIMARY
KEY (Nome_Genero_Pessoa);
ALTER TABLE filmes.Genero_Filme ADD CONSTRAINT PK_Genero_Filme PRIMARY KEY
(Nome_Genero_Filme);
ALTER TABLE filmes.Pessoa ADD CONSTRAINT PK_Pessoa PRIMARY KEY
(Nome_Pessoa);
ALTER TABLE filmes.Diretor ADD CONSTRAINT PK_Diretor PRIMARY KEY
(Nome_Pessoa);
ALTER TABLE filmes.Produtor ADD CONSTRAINT PK_Produtor PRIMARY KEY
(Nome_Pessoa);
ALTER TABLE filmes.Ator ADD CONSTRAINT PK_Ator PRIMARY KEY (Nome_Pessoa);
ALTER TABLE filmes.Plataforma ADD CONSTRAINT PK_Plataforma PRIMARY KEY
(Nome_Plataforma);
ALTER TABLE filmes.Site ADD CONSTRAINT PK_Site PRIMARY KEY (Nome_Site);
ALTER TABLE filmes.Premio ADD CONSTRAINT PK_Premio PRIMARY KEY
(Nome_Premio);
ALTER TABLE filmes.Filme ADD CONSTRAINT PK_Filme PRIMARY KEY (Titulo);

ALTER TABLE filmes.Pessoa ADD CONSTRAINT FK_Pessoa_Pais FOREIGN KEY
(Nome_Pais) REFERENCES filmes.Pais(Nome_Pais);
ALTER TABLE filmes.Pessoa ADD CONSTRAINT FK_Pessoa_Genero_Pessoa FOREIGN
KEY (Nome_Genero_Pessoa) REFERENCES
filmes.Genero_Pessoa(Nome_Genero_Pessoa);
```

```
ALTER TABLE filmes.Diretor ADD CONSTRAINT FK_Diretor_Pessoa FOREIGN KEY
(Nome_Pessoa) REFERENCES filmes.Pessoa(Nome_Pessoa);
ALTER TABLE filmes.Produtor ADD CONSTRAINT FK_Produtor_Pessoa FOREIGN KEY
(Nome_Pessoa) REFERENCES filmes.Pessoa(Nome_Pessoa);
ALTER TABLE filmes.Ator ADD CONSTRAINT FK_Ator_Pessoa FOREIGN KEY
(Nome_Pessoa) REFERENCES filmes.Pessoa(Nome_Pessoa);

ALTER TABLE filmes.Filme ADD CONSTRAINT FK_Filme_Pais FOREIGN KEY
(Nome_Pais) REFERENCES filmes.Pais(Nome_Pais);

CREATE TABLE filmes.Cotacao (
    Titulo_Filme VARCHAR(255),
    Nome_Site VARCHAR(255),
    Nota INT CHECK (Nota BETWEEN 0 AND 100),
    PRIMARY KEY (Titulo_Filme, Nome_Site),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Site) REFERENCES filmes.Site(Nome_Site)
);

CREATE TABLE filmes.Indicacao (
    Titulo_Filme VARCHAR(255),
    Nome_Premio VARCHAR(255),
    Ano INT,
    PRIMARY KEY (Titulo_Filme, Nome_Premio, Ano),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Premio) REFERENCES filmes.Premio(Nome_Premio)
);

CREATE TABLE filmes.Premiacao (
    Titulo_Filme VARCHAR(255),
    Nome_Premio VARCHAR(255),
    Ano INT,
    PRIMARY KEY (Titulo_Filme, Nome_Premio, Ano),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Premio) REFERENCES filmes.Premio(Nome_Premio)
);

CREATE TABLE filmes.Filme_Plataforma (
    Titulo_Filme VARCHAR(255),
```

```
    Nome_Plataforma VARCHAR(255),
    PRIMARY KEY (Titulo_Filme, Nome_Plataforma),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Plataforma) REFERENCES
filmes.Plataforma(Nome_Plataforma)
);

CREATE TABLE filmes.Filme_Produtor (
    Titulo_Filme VARCHAR(255),
    Nome_Pessoa VARCHAR(255),
    PRIMARY KEY (Titulo_Filme, Nome_Pessoa),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Pessoa) REFERENCES filmes.Produtor(Nome_Pessoa)
);

CREATE TABLE filmes.Filme_Diretor (
    Titulo_Filme VARCHAR(255),
    Nome_Pessoa VARCHAR(255),
    PRIMARY KEY (Titulo_Filme, Nome_Pessoa),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Pessoa) REFERENCES filmes.Diretor(Nome_Pessoa)
);

CREATE TABLE filmes.Filme_Ator (
    Titulo_Filme VARCHAR(255),
    Nome_Pessoa VARCHAR(255),
    PRIMARY KEY (Titulo_Filme, Nome_Pessoa),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Pessoa) REFERENCES filmes.Ator(Nome_Pessoa)
);

CREATE TABLE filmes.Filme_Genero (
    Titulo_Filme VARCHAR(255),
    Nome_Genero_Filme VARCHAR(255),
    PRIMARY KEY (Titulo_Filme, Nome_Genero_Filme),
    FOREIGN KEY (Titulo_Filme) REFERENCES filmes.Filme(Titulo),
    FOREIGN KEY (Nome_Genero_Filme) REFERENCES
filmes.Genero_Filme(Nome_Genero_Filme)
);
```

EXEMPLO INSERÇÃO:

```
-- Um Sonho de Liberdade
-- Inserindo filme
INSERT INTO filmes.Filme (Titulo, Duracao, Ano_Lancamento, Sinopse,
Nome_Pais) VALUES
('Um Sonho de Liberdade', 142, 1994, 'Um Sonho de Liberdade conta a
história de Andy Dufresne, um banqueiro condenado por assassinato que é
enviado para a prisão de Shawshank. Lá, ele desenvolve uma amizade com o
colega preso Ellis "Red" Redding, lida com a brutalidade da vida na prisão
e encontra redenção por meio de atos de decência comuns.', 'United States
of America');

-- Cadastrando pessoas relacionadas ao filme
INSERT INTO filmes.Pessoa (Nome_Pessoa, Data_Nascimento, Nome_Pais,
Nome_Genero_Pessoa) VALUES
('Niki Marvin', '1943-09-15', 'United States of America', 'Feminino'),
('Frank Darabont', '1959-01-28', 'United States of America', 'Masculino'),
('Tim Robbins', '1958-10-16', 'United States of America', 'Masculino'),
('Morgan Freeman', '1937-06-01', 'United States of America', 'Masculino'),
('Bob Gunton', '1945-11-15', 'United States of America', 'Masculino');

INSERT INTO filmes.Diretor (Nome_Pessoa) VALUES
('Frank Darabont');

INSERT INTO filmes.Produtor (Nome_Pessoa, Empresa) VALUES
('Niki Marvin', 'Castle Rock Entertainment');

INSERT INTO filmes.Ator (Nome_Pessoa) VALUES
('Tim Robbins'),
('Morgan Freeman'),
('Bob Gunton');

-- Relacionando pessoas cadastradas ao filme
INSERT INTO filmes.Filme_Produtor (Titulo_Filme, Nome_Pessoa) VALUES
('Um Sonho de Liberdade', 'Niki Marvin');

INSERT INTO filmes.Filme_Diretor (Titulo_Filme, Nome_Pessoa) VALUES
```

```
('Um Sonho de Liberdade', 'Frank Darabont');

INSERT INTO filmes.Filme_Ator (Titulo_Filme, Nome_Pessoa) VALUES
('Um Sonho de Liberdade', 'Tim Robbins'),
('Um Sonho de Liberdade', 'Morgan Freeman'),
('Um Sonho de Liberdade', 'Bob Gunton');

-- Relacionar às plataformas que o disponibilizam
INSERT INTO filmes.Filme_Plataforma (Titulo_Filme, Nome_Plataforma) VALUES
('Um Sonho de Liberdade', 'Netflix'),
('Um Sonho de Liberdade', 'Amazon Prime Video');

-- Relacionando gêneros do filme
INSERT INTO filmes.Filme_Genero (Titulo_Filme, Nome_Genero_Filme) VALUES
('Um Sonho de Liberdade', 'Drama'),
('Um Sonho de Liberdade', 'Crime');

-- Relacionando cotações ao filme
INSERT INTO filmes.Cotacao (Titulo_Filme, Nome_Site, Nota) VALUES
('Um Sonho de Liberdade', 'IMDb', 93),
('Um Sonho de Liberdade', 'Rotten Tomatoes', 91),
('Um Sonho de Liberdade', 'Metacritic', 80),
('Um Sonho de Liberdade', 'Letterboxd', 90),
('Um Sonho de Liberdade', 'FilmAffinity', 83),
('Um Sonho de Liberdade', 'The Movie Database (TMDb)', 87);

-- Relacionando indicações cadastradas ao filme
INSERT INTO filmes.Indicacao (Titulo_Filme, Nome_Premio, Ano) VALUES
('Um Sonho de Liberdade', 'Oscar - Melhor Filme', 1994),
('Um Sonho de Liberdade', 'Oscar - Melhor Ator', 1994),
('Um Sonho de Liberdade', 'Oscar - Melhor Roteiro Adaptado', 1994),
('Um Sonho de Liberdade', 'Oscar - Melhor Fotografia', 1994),
('Um Sonho de Liberdade', 'Oscar - Melhor Edição', 1994),
('Um Sonho de Liberdade', 'Oscar - Melhor Trilha Sonora Original', 1994),
('Um Sonho de Liberdade', 'Oscar - Melhor Som', 1994),
('Um Sonho de Liberdade', 'Globo de Ouro - Melhor Ator - Drama', 1994),
('Um Sonho de Liberdade', 'Globo de Ouro - Melhor Roteiro', 1994),
('Um Sonho de Liberdade', 'BAFTA - Melhor Roteiro Adaptado', 1994);
```

QUESTÃO 7/8

IDEIAS

1. Análise de Distribuição Global de Filmes:
    -Objetivo: Investigar a distribuição global da produção cinematográfica, identificando os países que produzem mais filmes.
    -Dados Utilizados: Tabelas Filme e Pais.
    -Métodos: Junção e agregação para combinar dados e determinar tendências de produção por região.

2. Diversidade de Gênero na Indústria Cinematográfica:
    -Objetivo: Analisar a representatividade de gêneros entre atores, diretores e produtores.
    -Dados Utilizados: Tabelas Pessoa, Genero_Pessoa, Diretor, Ator.
    -Métodos: Junções para revelar insights sobre diversidade e inclusão.

3. Correlação entre Plataformas de Distribuição e Sucesso de Filmes:
    -Objetivo: Avaliar se a disponibilidade de filmes em múltiplas plataformas está correlacionada com melhores avaliações ou mais premiações.
    -Dados Utilizados: Tabelas Filme_Plataforma, Cotacao, Premiacao.

-Métodos: Junções e cruzamento de informações.

4. Impacto da Nacionalidade do Diretor no Sucesso Internacional do Filme:
   -Objetivo: Investigar se a nacionalidade dos diretores afeta o sucesso internacional dos filmes.
   -Dados Utilizados: Tabelas Diretor, Pessoa, Pais, Filme.
   -Métodos: Combinação de dados para observar padrões de aceitação em diferentes culturas.

5. Tendências Históricas em Gêneros de Filmes:
   -Objetivo: Explorar as mudanças nas preferências dos gêneros cinematográficos ao longo das décadas.
   -Dados Utilizados: Tabela Filme_Generos.
   -Métodos: Agregações por ano e gênero para identificar tendências emergentes e decrescentes.

6. Análise de Colaboração entre Profissionais da Indústria:
   -Objetivo: Avaliar a frequência e o impacto das colaborações entre diretores e atores.
   -Dados Utilizados: Tabelas Filme_Diretores, Filme_Atores, Pessoa, Diretor, Ator.
   -Métodos: Junções para examinar como essas colaborações influenciam a recepção crítica e comercial.

7. Comparação de Avaliações entre Crítica e Público:
   -Objetivo: Comparar a percepção da crítica com a avaliação do público para os filmes.

-Dados Utilizados: Tabelas Cotacao, Indicacao, Premiacao.
-Métodos: Integração de dados e utilização de operações como diferença e projeção.

8. Análise dos Produtores e Sucesso dos Filmes:
-Objetivo: Avaliar a influência dos produtores no sucesso dos filmes em termos de premiações.
-Dados Utilizados: Tabelas Produtor, Pessoa, Filme_Produtores, Premiacao.
-Métodos: Junções para analisar a correlação entre produtores e premiações recebidas.

9. Análise de Filmes por Gênero e Premiações:
-Objetivo: Investigar a relação entre os gêneros de filmes e as premiações que recebem.
-Dados Utilizados: Tabelas Filme_Generos, Premiacao.
-Métodos: Junções e análise de dados para entender a frequência de premiações por gênero.

10. Distribuição de Filmes por Sites e Avaliações:
-Objetivo: Explorar a relação entre a visibilidade de filmes em diferentes sites e suas avaliações.
-Dados Utilizados: Tabelas Filme, Site, Cotacao.
-Métodos: Junções e análise de médias para correlacionar visibilidade e avaliações.

11. Filmes Indicados a Premiações por Ano:

-Objetivo: Identificar filmes indicados a diferentes premiações ao longo dos anos.
-Dados Utilizados: Tabelas Filme, Premio, Indicacao.
-Métodos: Junções para listar filmes e suas respectivas indicações por ano.

12. Impacto da Duração do Filme nas Avaliações:
-Objetivo: Analisar como a duração dos filmes influencia suas avaliações e premiações.
-Dados Utilizados: Tabelas Filme, Cotacao, Premiacao.
-Métodos: Junções e análise de correlação para verificar se existe uma relação entre a duração dos filmes e suas avaliações ou premiações.

13. Análise de Premiação por Nacionalidade de Produtores:
-Objetivo: Avaliar se há uma correlação entre a nacionalidade dos produtores e a quantidade de prêmios que seus filmes recebem.
-Dados Utilizados: Tabelas Produtor, Pessoa, Pais, Filme, Premiacao.
-Métodos: Junções e análises de correlação para investigar se a nacionalidade dos produtores influencia o sucesso em termos de premiações.

Escolhidas - Algebra Relacional // SQL

8. Análise dos Produtores e Sucesso dos Filmes:

Algebra:

ProdutoresPessoas <- σ (produtor.Nome_Pessoa = pessoa.Nome_Pessoa) (produtor × pessoa)

FilmesProdutoresPessoas <- σ (filme_produtor.Nome_Pessoa = produtor.Nome_Pessoa) (filme_produtor × ProdutoresPessoas)

FilmesProdutoresDetalhes <- σ (filme.Titulo = filme_produtor.Titulo_Filme) (FilmesProdutoresPessoas × filme)

FilmesProdutoresPremiacoes <- (FilmesProdutoresDetalhes ⋈ premiacao.Titulo_Filme = filme.Titulo)

Resultado <- π (Nome_Pessoa, Empresa, Titulo, Nome_Premio) (FilmesProdutoresPremiacoes)

TotalPremiacoes <- γ (Nome_Pessoa, Empresa, Titulo; COUNT(Nome_Premio)) (Resultado)

SQL:

```sql
SELECT
    p.nome_pessoa AS nome_produtor,
    pr.empresa,
    f.titulo AS titulo_filme,
    COUNT(pm.nome_premio) AS total_premiacoes
FROM
    filmes.produtor pr
    JOIN filmes.pessoa p ON pr.nome_pessoa = p.nome_pessoa
    JOIN filmes.filme_produtor fp ON pr.nome_pessoa = fp.nome_pessoa
    JOIN filmes.filme f ON fp.titulo_filme = f.titulo
```

```
    LEFT JOIN filmes.premiacao pm ON f.titulo = pm.titulo_filme
GROUP BY
    p.nome_pessoa, pr.empresa, f.titulo
ORDER BY
    total_premiacoes DESC;
```

10. Distribuição de Filmes por Sites e Avaliações:

Algebra:

FilmesCotacoes <- σ (filme.titulo = cotacao.titulo_filme) (filme × cotacao)

MediaAvaliacoes <- γ (cotacao.titulo_filme, cotacao.nome_site; AVG(cotacao.nota)) (FilmesCotacoes)

MediaAvaliacoesSites <- σ (MediaAvaliacoes.nome_site = site.nome_site) (MediaAvaliacoes × site)

Resultado <- σ (MediaAvaliacoes.titulo_filme = filme.titulo) (MediaAvaliacoesSites × filme)

ResultadoFinal <- π (filme.titulo, site.nome_site, site.url, MediaAvaliacoes.media_avaliacao) (Resultado)

SQL:

```
SELECT
    f.titulo,
```

```
    s.nome_site,
    s.url,
    ROUND(Filme_Site_Cotacao.media_avaliacao, 2) AS media_avaliacao
FROM
    (SELECT
        c.titulo_filme,
        c.nome_site,
        AVG(c.nota) AS media_avaliacao
     FROM
        filmes.cotacao c
     GROUP BY
        c.titulo_filme,
        c.nome_site) AS Filme_Site_Cotacao
JOIN
    filmes.site s
ON
    Filme_Site_Cotacao.nome_site = s.nome_site
JOIN
    filmes.filme f
ON
    Filme_Site_Cotacao.titulo_filme = f.titulo
ORDER BY
    media_avaliacao DESC;
```

6. Análise de Colaboração entre Profissionais da Indústria:

Algebra:

FilmesDiretores <- σ (filme_diretor.titulo_filme = filme.titulo) (filme_diretor × filme)

FilmesDiretoresDetalhes <- σ (filme_diretor.nome_pessoa = diretor.nome_pessoa) (FilmesDiretores × diretor)

FilmesDiretoresPessoas <- σ (diretor.nome_pessoa = pessoa.nome_pessoa) (FilmesDiretoresDetalhes × pessoa)

FilmesAtores <- σ (filme_ator.titulo_filme = filme.titulo) (filme_ator × filme)

FilmesAtoresDetalhes <- σ (filme_ator.nome_pessoa = ator.nome_pessoa) (FilmesAtores × ator)

FilmesAtoresPessoas <- σ (ator.nome_pessoa = pessoa.nome_pessoa) (FilmesAtoresDetalhes × pessoa)

Colaboracoes <- σ (FilmesDiretoresPessoas.titulo_filme = FilmesAtoresPessoas.titulo_filme) (FilmesDiretoresPessoas × FilmesAtoresPessoas)

ColaboracoesCotacoes <- (Colaboracoes ⋈ cotacao.titulo_filme = FilmesDiretoresPessoas.titulo_filme)

ColaboracoesCompleta <- (ColaboracoesCotacoes ⋈ premiacao.titulo_filme = FilmesDiretoresPessoas.titulo_filme)

Resultado <- π (FilmesDiretoresPessoas.nome_pessoa, FilmesAtoresPessoas.nome_pessoa, FilmesDiretoresPessoas.titulo_filme, cotacao.nota, premiacao.titulo_filme) (ColaboracoesCompleta)

Estatisticas <- γ (FilmesDiretoresPessoas.nome_pessoa, FilmesAtoresPessoas.nome_pessoa; COUNT(DISTINCT FilmesDiretoresPessoas.titulo_filme), AVG(cotacao.nota), COUNT(premiacao.titulo_filme)) (Resultado)

SQL:

```
SELECT
    d.nome_pessoa AS nome_diretor,
```

```
    a.nome_pessoa AS nome_ator,
    COUNT(DISTINCT f.titulo) AS numero_colaboracoes,
    ROUND(AVG(c.nota), 2) AS media_critica,
    COUNT(p.titulo_filme) AS numero_premiacoes
FROM
    filmes.filme_diretor fd
JOIN
    filmes.filme f ON fd.titulo_filme = f.titulo
JOIN
    filmes.diretor di ON fd.nome_pessoa = di.nome_pessoa
JOIN
    filmes.pessoa d ON di.nome_pessoa = d.nome_pessoa
JOIN
    filmes.filme_ator fa ON f.titulo = fa.titulo_filme
JOIN
    filmes.ator at ON fa.nome_pessoa = at.nome_pessoa
JOIN
    filmes.pessoa a ON at.nome_pessoa = a.nome_pessoa
LEFT JOIN
    filmes.cotacao c ON f.titulo = c.titulo_filme
LEFT JOIN
    filmes.premiacao p ON f.titulo = p.titulo_filme
GROUP BY
    d.nome_pessoa, a.nome_pessoa
ORDER BY
    numero_colaboracoes DESC, media_critica DESC, numero_premiacoes DESC;
```

9. Análise de Filmes por Gênero e Premiações:

Algebra:

GeneroFilmeGeneros <- σ (genero_filme.nome_genero_filme = filme_genero.nome_genero_filme) (genero_filme × filme_genero)

GeneroFilmeDetalhes <- σ (filme_genero.titulo_filme = filme.titulo) (GeneroFilmeGeneros × filme)

GeneroFilmePremiacoes <- σ (filme.titulo = premiacao.titulo_filme) (GeneroFilmeDetalhes × premiacao)

Resultado <- π (genero_filme.nome_genero_filme, premiacao.nome_premio) (GeneroFilmePremiacoes)

TotalPremiosPorGenero <- γ (genero_filme.nome_genero_filme; COUNT(premiacao.nome_premio)) (Resultado)

SQL:

```sql
SELECT
    gf.nome_genero_filme AS genero,
    COUNT(p.nome_premio) AS total_premios
FROM
    filmes.genero_filme gf
JOIN
    filmes.filme_genero fg ON gf.nome_genero_filme = fg.nome_genero_filme
JOIN
    filmes.filme f ON fg.titulo_filme = f.titulo
JOIN
    filmes.premiacao p ON f.titulo = p.titulo_filme
GROUP BY
    gf.nome_genero_filme;
```

2. Diversidade de Gênero na Indústria Cinematográfica:

Algebra:

GeneroPessoaDetalhes <- σ (genero_pessoa.nome_genero_pessoa = pessoa.nome_genero_pessoa) (genero_pessoa × pessoa)

Resultado <- π (genero_pessoa.nome_genero_pessoa, pessoa.nome_pessoa) (GeneroPessoaDetalhes)

TotalPessoasPorGenero <- γ (genero_pessoa.nome_genero_pessoa; COUNT(pessoa.nome_pessoa)) (Resultado)

SQL:

```sql
SELECT
    gp.nome_genero_pessoa AS genero,
    COUNT(p.nome_pessoa) AS total_pessoas
FROM
    filmes.genero_pessoa gp
JOIN
    filmes.pessoa p ON gp.nome_genero_pessoa = p.nome_genero_pessoa
GROUP BY
    gp.nome_genero_pessoa;
```

12. Filmes Indicados a Premiações por Ano:

Algebra:

FilmesIndicacoes <- σ (filme.titulo = indicacao.titulo_filme) (filme × indicacao)

FilmesIndicacoesPremios <- σ (indicacao.nome_premio = premio.nome_premio) (FilmesIndicacoes × premio)

Resultado <- π (filme.titulo, premio.nome_premio, indicacao.ano) (FilmesIndicacoesPremios)

ResultadoFinal <- δ (Resultado)

SQL:

```sql
SELECT DISTINCT
    f.titulo,
    p.nome_premio,
    i.ano
FROM
    filmes.filme f
JOIN
    filmes.indicacao i ON f.titulo = i.titulo_filme
JOIN
    filmes.premio p ON i.nome_premio = p.nome_premio
ORDER BY
    i.ano,
    f.titulo,
    p.nome_premio;
```

Escolhidas - SQL

1. Análise de Distribuição Global de Filmes

```sql
SELECT
    p.nome_pais AS nome_pais,
    COUNT(f.titulo) AS numero_filmes
FROM
    filmes.filme f
JOIN
    filmes.pais p ON f.nome_pais = p.nome_pais
GROUP BY
    p.nome_pais
ORDER BY
    numero_filmes DESC;
```

4. Impacto da Nacionalidade do Diretor no Sucesso Internacional do Filme

```sql
SELECT
    p.nome_pais AS pais,
    COUNT(pr.nome_premio) AS total_premios
FROM
    filmes.filme f
JOIN
    filmes.filme_diretor fd ON f.titulo = fd.titulo_filme
JOIN
    filmes.diretor d ON fd.nome_pessoa = d.nome_pessoa
JOIN
    filmes.pessoa ps ON d.nome_pessoa = ps.nome_pessoa
JOIN
    filmes.pais p ON ps.nome_pais = p.nome_pais
JOIN
    filmes.premiacao pr ON f.titulo = pr.titulo_filme
GROUP BY
    p.nome_pais
ORDER BY
    total_premios DESC;
```

5. Tendências Históricas em Gêneros de Filmes

```sql
SELECT
    f.ano_lancamento AS ano,
    gf.nome_genero_filme AS genero,
    COUNT(fg.titulo_filme) AS total_filmes
FROM
    filmes.filme f
JOIN
    filmes.filme_genero fg ON f.titulo = fg.titulo_filme
JOIN
    filmes.genero_filme gf ON fg.nome_genero_filme = gf.nome_genero_filme
GROUP BY
    f.ano_lancamento, gf.nome_genero_filme
ORDER BY
    f.ano_lancamento, total_filmes DESC;
```

3. Correlação entre Plataformas de Distribuição e Sucesso de Filmes

```sql
SELECT
    f.titulo AS nome_filme,
    COUNT(DISTINCT fp.nome_plataforma) AS numero_plataformas,
    ROUND(AVG(c.nota), 2) AS media_avaliacao,
    COUNT(DISTINCT pr.nome_premio) AS total_premios
FROM
    filmes.filme f
LEFT JOIN
    filmes.filme_plataforma fp ON f.titulo = fp.titulo_filme
LEFT JOIN
    filmes.cotacao c ON f.titulo = c.titulo_filme
LEFT JOIN
    filmes.premiacao pr ON f.titulo = pr.titulo_filme
GROUP BY
    f.titulo
ORDER BY
    numero_plataformas DESC, media_avaliacao DESC, total_premios DESC;
```

Questão 10

1. Média de Avaliações por Plataforma:

Ideia: Calcular a média das avaliações dos filmes disponíveis em cada plataforma.

SQL:

```sql
SELECT fp.Nome_Plataforma, ROUND(AVG(c.Nota),0) AS Media_Avaliacao
FROM Filme f
JOIN Cotacao c ON f.Titulo = c.Titulo_Filme
JOIN Filme_Plataforma fp ON f.Titulo = fp.Titulo_Filme
GROUP BY fp.Nome_Plataforma;
```

2. Número de Filmes por Gênero e País

Ideia: Contar o número de filmes de cada gênero por país.

SQL:

```sql
SELECT p.Nome_Pais, fg.Nome_Genero_Filme, COUNT(f.Titulo) AS Numero_Filmes
FROM Filme f
JOIN Filme_Genero fg ON f.Titulo = fg.Titulo_Filme
JOIN Pais p ON f.Nome_Pais = p.Nome_Pais
GROUP BY p.Nome_Pais, fg.Nome_Genero_Filme;
```

3.Diretores com Mais Prêmios

Ideia: Listar, em ordem decrescente, os diretores que dirigiram filmes que ganharam mais prêmios, mostrar o nome do diretor, o título do filme e a contagem de prêmios que cada filme ganhou.

SQL:

```sql
SELECT p.Nome_Pessoa AS Nome_Diretor, f.Titulo AS Titulo_Filme,
COUNT(pr.Nome_Premio) AS Numero_Premios
FROM Filme f
```

```
JOIN Filme_Diretor fd ON f.Titulo = fd.Titulo_Filme
JOIN Premiacao pr ON f.Titulo = pr.Titulo_Filme
JOIN Pessoa p ON fd.Nome_Pessoa = p.Nome_Pessoa
GROUP BY p.Nome_Pessoa, f.Titulo
ORDER BY Numero_Premios DESC;
```

4.Filmes com Avaliação Média Superior 80

Ideia: Encontrar filmes que possuem uma média de avaliações superior a 80 e ordená-los em ordem decrescente.

SQL:

```
SELECT f.Titulo, ROUND(AVG(c.Nota),0) AS Media_Avaliacao
FROM Filme f
JOIN Cotacao c ON f.Titulo = c.Titulo_Filme
GROUP BY f.Titulo
HAVING AVG(c.Nota) > 80
ORDER BY AVG(c.Nota) DESC;
```

5.Classificação de Filmes com Base nas Avaliações

Ideia: Criar uma consulta que classifique os filmes em diferentes categorias ("Excelente", "Bom", "Regular" e "Ruim") com base na média de suas avaliações.

SQL:

```
SELECT f.Titulo, ROUND(AVG(c.Nota),0) AS Media_Avaliacao,
       CASE
           WHEN AVG(c.Nota) > 90 THEN 'Excelente'
           WHEN AVG(c.Nota) BETWEEN 70 AND 90 THEN 'Bom'
           WHEN AVG(c.Nota) BETWEEN 50 AND 70 THEN 'Regular'
           ELSE 'Ruim'
       END AS Classificacao
FROM Filme f
JOIN Cotacao c ON f.Titulo = c.Titulo_Filme
GROUP BY f.Titulo
ORDER BY Media_Avaliacao DESC;
```

6.Filmes com a Maior Duração por Gênero
Ideia: Criar uma consulta que encontre o filme com a maior duração em cada gênero.

SQL:

```sql
WITH Max_Duration AS (
    SELECT fg.Nome_Genero_Filme, MAX(f.Duracao) AS Max_Duracao
    FROM Filme f
    JOIN Filme_Genero fg ON f.Titulo = fg.Titulo_Filme
    GROUP BY fg.Nome_Genero_Filme
)

SELECT fg.Nome_Genero_Filme, f.Titulo, f.Duracao
FROM Filme f
JOIN Filme_Genero fg ON f.Titulo = fg.Titulo_Filme
JOIN Max_Duration md ON fg.Nome_Genero_Filme = md.Nome_Genero_Filme AND
f.Duracao = md.Max_Duracao
ORDER BY fg.Nome_Genero_Filme;
```

Questão 11 (A)

1.1 Filmes e Suas Plataformas de Distribuição
Objetivo: Criar uma visão que liste todos os filmes e as plataformas onde estão disponíveis.

SQL - Visão:

```sql
CREATE VIEW Filmes_Plataformas AS
SELECT f.Titulo, fp.Nome_Plataforma
FROM Filme f
JOIN Filme_Plataforma fp ON f.Titulo = fp.Titulo_Filme;
```

1.2 Liste os filmes que estão disponíveis em mais de duas plataformas de distribuição.

SQL - Exemplo:

```sql
SELECT Titulo, COUNT(Nome_Plataforma) AS Numero_Plataformas
FROM Filmes_Plataformas
GROUP BY Titulo
HAVING COUNT(Nome_Plataforma) > 2;
```

2.1 Filmes por País com Avaliação Média

Objetivo: Criar uma visão que liste todos os filmes, seus países de origem e a média das suas avaliações.

SQL - Visão:

```sql
CREATE VIEW Filmes_Por_Pais_Avaliacao AS
SELECT f.Titulo, f.Nome_Pais, ROUND(AVG(c.Nota), 0) AS Media_Avaliacao
FROM Filme f
JOIN Cotacao c ON f.Titulo = c.Titulo_Filme
GROUP BY f.Titulo, f.Nome_Pais;
```

2.2 Encontre todos os filmes Estado Unidenses com média de avaliação maior que 80.

SQL - Exemplo:

```sql
SELECT Titulo, Media_Avaliacao
FROM Filmes_Por_Pais_Avaliacao
WHERE Nome_Pais = 'United States of America' AND Media_Avaliacao > 80;
```

Questão 11 (B)

1.Filmes Estado Unidenses

Objetivo: Criar uma visão que liste apenas filmes dos Estados Unidos.

SQL - Visão:

```sql
CREATE VIEW Filmes_Estado_Unidenses AS
SELECT Titulo, Ano_Lancamento, Sinopse, Nome_Pais
FROM Filme
WHERE Nome_Pais = 'United States of America'
WITH CHECK OPTION;
```

SQL - Sucesso:

```sql
UPDATE Filmes_Estado_Unidenses
SET Sinopse = 'Nova Sinopse'
WHERE Titulo = 'O Poderoso Chefão';
```

Antes:

Editar	Deletar	O Poderoso Chefão	175	1972	A saga da família Corleone, uma das mais poderos...	United States of America

Depois:

Editar	Deletar	O Poderoso Chefão	175	1972	Nova Sinopse	United States of America

SQL - Falha:

```sql
UPDATE Filmes_Estado_Unidenses
SET Nome_Pais = 'Argentina'
WHERE Titulo = 'Forrest Gump';
```

Erro:

Erro de SQL:

```
ERROR:  new row violates check option for view "filmes_estado_unidenses"
DETAIL:  Failing row contains (Forrest Gump, 142, 1994, Forrest Gump, um homem com QI baixo mas
```

No bloco:

```
UPDATE Filmes_Estado_Unidenses
SET Nome_Pais = 'Argentina'
WHERE Titulo = 'Forrest Gump';
```

2.Filmes com Duração Longa

Objetivo: Criar uma visão que liste apenas filmes com duração superior a 120 minutos.

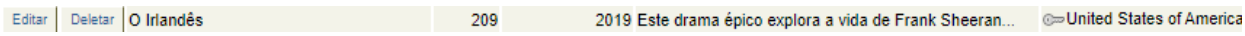
SQL - Visão:

```sql
CREATE VIEW Filmes_Duracao_Longa AS
SELECT Titulo, Ano_Lancamento, Sinopse, Nome_Pais, Duracao
FROM Filme
WHERE Duracao > 120
WITH CHECK OPTION;
```

SQL - Sucesso:

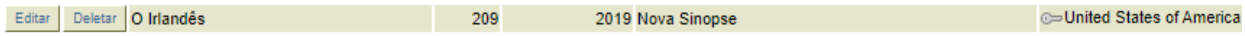
```sql
UPDATE Filmes_Duracao_Longa
SET Sinopse = 'Nova Sinopse'
WHERE Titulo = 'O Irlandês';
```

Antes:



Editar	Deletar	O Irlandês	209	2019	Este drama épico explora a vida de Frank Sheeran...	United States of America

Depois:



Editar	Deletar	O Irlandês	209	2019	Nova Sinopse	United States of America

SQL - Falha:

```sql
UPDATE Filmes_Duracao_Longa
SET Duracao = 110
WHERE Titulo = 'Matrix';
```

Erro:

Erro de SQL:

```
ERROR:  new row violates check option for view "filmes_duracao_longa"
DETAIL:  Failing row contains (Matrix, 110, 1999, Um hacker conhecido como Neo descobre que
```

No bloco:

```sql
UPDATE Filmes_Duracao_Longa
SET Duracao = 110
WHERE Titulo = 'Matrix';
```

Questão 12 (A)

1.Função para Calcular a Média de Avaliação de um Filme

SQL - Função:

```
CREATE FUNCTION calcula_media_avaliacao(titulo_filme_calc VARCHAR) RETURNS
FLOAT AS $$
DECLARE
    media_avaliacao FLOAT;
BEGIN
    SELECT ROUND(AVG(nota)) INTO media_avaliacao
    FROM Cotacao
    WHERE titulo_filme = titulo_filme_calc;
    RETURN media_avaliacao;
END;
$$ LANGUAGE plpgsql;
```

SQL - Pesquisa:

```
SELECT
    calcula_media_avaliacao('Parasita') AS AV_Parasita,
    calcula_media_avaliacao('O Irlandês') AS AV_OIrlandês,
    calcula_media_avaliacao('Casablanca') AS AV_OCasablanca,
    calcula_media_avaliacao('O Rei Leão') AS AV_OReiLeão;
```

Resultado:

av_parasita	av_oirlandês	av_ocasablanca	av_oreileão
90	83	91	86

2.Função para Contar o Número de Filmes por Diretor

SQL - Função:

```
CREATE FUNCTION conta_filmes_por_diretor(nome_diretor VARCHAR) RETURNS
INTEGER AS $$
DECLARE
    num_filmes INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO num_filmes
    FROM Filme f
    JOIN Filme_Diretor fd ON f.Titulo = fd.Titulo_Filme
```

```
    WHERE fd.Nome_Pessoa = nome_diretor;
    RETURN num_filmes;
END;
$$ LANGUAGE plpgsql;
```

SQL - Pesquisa:

```
SELECT
 conta_filmes_por_diretor('Christopher Nolan') AS D_Christopher ,
 conta_filmes_por_diretor('Francis Ford Coppola') AS D_Francis ,
 conta_filmes_por_diretor('Robert Zemeckis') AS D_Robert;
```

Resultado:

d_christopher	d_francis	d_robert
3	3	2

Questão 12 (B)

Atualização e Inserção de Cotações de Filmes

Objetivo: Criar uma Stored Procedure que permita adicionar ou atualizar cotações de filmes em sites de avaliação. Este procedimento garante que a cotação de um filme seja inserida se não existir e atualizada se já estiver presente, mantendo a integridade e consistência dos dados no banco de dados.

Justificativa: A Stored Procedure adiciona_cotacao foi desenvolvida para garantir a eficiente gestão das cotações de filmes dentro do banco de dados. Esta procedure permite a inserção e atualização de cotações, assegurando que as avaliações mais recentes e precisas estejam sempre disponíveis.

SQL - SP:

```
CREATE OR REPLACE FUNCTION adiciona_cotacao(
```

```
    p_titulo_filme VARCHAR,
    p_nome_site VARCHAR,
    p_nota INTEGER
) RETURNS VOID AS $$
BEGIN
    -- Verifica se a cotação já existe
    IF EXISTS (SELECT 1 FROM Cotacao WHERE Titulo_Filme = p_titulo_filme
AND Nome_Site = p_nome_site) THEN
        -- Atualiza a cotação existente
        UPDATE Cotacao
        SET Nota = p_nota
        WHERE Titulo_Filme = p_titulo_filme AND Nome_Site = p_nome_site;
    ELSE
        -- Insere uma nova cotação
        INSERT INTO Cotacao (Titulo_Filme, Nome_Site, Nota)
        VALUES (p_titulo_filme, p_nome_site, p_nota);
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Exemplo - Teste: Vamos supor que os filmes “Parasita”, “O Irlandês” e “O Rei Leão”, no site “Rotten Tomatoes” diminuíram um ponto (processo manual no exemplo).

SQL - Consulta (Estado inicial):

```
SELECT * FROM Cotacao WHERE Titulo_Filme IN ('Parasita', 'O Irlandês', 'O
Rei Leão') AND Nome_Site = 'Rotten Tomatoes';
```

titulo_filme	nome_site	nota
Parasita	Rotten Tomatoes	99
O Irlandês	Rotten Tomatoes	96
O Rei Leão	Rotten Tomatoes	93

SQL - Modificar cotações com “adiciona_cotacao”:

```
SELECT adiciona_cotacao('Parasita', 'Rotten Tomatoes', 98);
SELECT adiciona_cotacao('O Irlandês', 'Rotten Tomatoes', 95);
SELECT adiciona_cotacao('O Rei Leão', 'Rotten Tomatoes', 92);
```

SQL - Consulta (Estado final):

```
SELECT * FROM Cotacao WHERE Titulo_Filme IN ('Parasita', 'O Irlandês', 'O
Rei Leão') AND Nome_Site = 'Rotten Tomatoes';
```

titulo_filme	nome_site	nota
Parasita	Rotten Tomatoes	98
O Irlandês	Rotten Tomatoes	95
O Rei Leão	Rotten Tomatoes	92

Questão 12 (C)

Remoção Automática de Pessoas Associadas ao Excluir um Filme

Objetivo: Criar um gatilho que assegure que todas as referências a um filme nas tabelas Filme_Ator, Filme_Diretor e Filme_Produtor sejam verificadas. Se as pessoas associadas não estiverem associadas a outros filmes, elas serão removidas da tabela Pessoa, para que não existam pessoas que não são ligadas a nenhum filme.

Sub-Objetivo: Além disso essa função também limpa todas as coisas que estão relacionadas com filme de maneira direta (que geram erro com minhas configurações de delete atuais, que preferi manter)

*Ela exclui os dados diretamente relacionados ao filme (com restrição no delete) e depois verifica pessoa a pessoa em

Diretor/Ator/Produtor ligadas ao filme, se x pessoa só estiver relacionada a um filme então ela apaga o cadastro geral da pessoa no banco de dados, se não ela prossegue e apenas exclui em Filme_Diretor/Ator/Produtor.

SQL - Função auxiliar (saber a quantos filmes essa pessoa está associada):

```
CREATE OR REPLACE FUNCTION conta_obras_por_pessoa(p_nome_pessoa VARCHAR)
RETURNS INTEGER AS $$
DECLARE
    num_obras INTEGER;
BEGIN
    SELECT COUNT(DISTINCT Titulo_Filme) INTO num_obras
    FROM (
        SELECT Titulo_Filme FROM Filme_Ator WHERE Nome_Pessoa =
p_nome_pessoa
        UNION
        SELECT Titulo_Filme FROM Filme_Diretor WHERE Nome_Pessoa =
p_nome_pessoa
        UNION
        SELECT Titulo_Filme FROM Filme_Produtor WHERE Nome_Pessoa =
p_nome_pessoa
    ) AS Obras;

    RETURN num_obras;
END;
$$ LANGUAGE plpgsql;
```

SQL - Funçao do trigger:

```
CREATE OR REPLACE FUNCTION verifica_e_remove_pessoas_e_referencias()
RETURNS TRIGGER AS $$
DECLARE
    pessoa RECORD;
BEGIN
    -- Remover cotações associadas ao filme excluído
    DELETE FROM Cotacao WHERE Titulo_Filme = OLD.Titulo;
```

```
    -- Remover indicações associadas ao filme excluído
    DELETE FROM Indicacao WHERE Titulo_Filme = OLD.Titulo;

    -- Remover premiações associadas ao filme excluído
    DELETE FROM Premiacao WHERE Titulo_Filme = OLD.Titulo;

    -- Remover filme_plataforma associadas ao filme excluído
    DELETE FROM Filme_Plataforma WHERE Titulo_Filme = OLD.Titulo;

    -- Remover filme_genero associadas ao filme excluído
    DELETE FROM Filme_Genero WHERE Titulo_Filme = OLD.Titulo;

    -- Verificar e remover diretores associados ao filme excluído //
pessoas
    FOR pessoa IN (SELECT Nome_Pessoa FROM Filme_Diretor WHERE
Titulo_Filme = OLD.Titulo)
    LOOP
        IF conta_obras_por_pessoa(pessoa.Nome_Pessoa) = 1 THEN
            DELETE FROM Filme_Diretor fd WHERE fd.Nome_Pessoa =
pessoa.Nome_Pessoa AND Titulo_Filme = OLD.Titulo;
            DELETE FROM Diretor d WHERE d.Nome_Pessoa =
pessoa.Nome_Pessoa;
            DELETE FROM Pessoa p WHERE p.Nome_Pessoa = pessoa.Nome_Pessoa;
        ELSE
            DELETE FROM Filme_Diretor fd WHERE fd.Nome_Pessoa =
pessoa.Nome_Pessoa AND Titulo_Filme = OLD.Titulo;
        END IF;
    END LOOP;

    -- Verificar e remover atores associados ao filme excluído // pessoas
    FOR pessoa IN (SELECT Nome_Pessoa FROM Filme_Ator WHERE Titulo_Filme =
OLD.Titulo)
    LOOP
        IF conta_obras_por_pessoa(pessoa.Nome_Pessoa) = 1 THEN
            DELETE FROM Filme_Ator fa WHERE fa.Nome_Pessoa =
pessoa.Nome_Pessoa AND Titulo_Filme = OLD.Titulo;
            DELETE FROM Ator a WHERE a.Nome_Pessoa = pessoa.Nome_Pessoa;
            DELETE FROM Pessoa p WHERE p.Nome_Pessoa = pessoa.Nome_Pessoa;
        ELSE
```

```
            DELETE FROM Filme_Ator fa WHERE fa.Nome_Pessoa =
pessoa.Nome_Pessoa AND Titulo_Filme = OLD.Titulo;
        END IF;
    END LOOP;

    -- Verificar e remover produtores associados ao filme excluído //
pessoas
    FOR pessoa IN (SELECT Nome_Pessoa FROM Filme_Produtor WHERE
Titulo_Filme = OLD.Titulo)
    LOOP
        IF conta_obras_por_pessoa(pessoa.Nome_Pessoa) = 1 THEN
            DELETE FROM Filme_Produtor fp WHERE fp.Nome_Pessoa =
pessoa.Nome_Pessoa AND Titulo_Filme = OLD.Titulo;
            DELETE FROM Produtor pr WHERE pr.Nome_Pessoa =
pessoa.Nome_Pessoa;
            DELETE FROM Pessoa p WHERE Nome_Pessoa = pessoa.Nome_Pessoa;
        ELSE
            DELETE FROM Filme_Produtor WHERE Nome_Pessoa =
pessoa.Nome_Pessoa AND Titulo_Filme = OLD.Titulo;
        END IF;
    END LOOP;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

SQL - Trigger:

```
CREATE TRIGGER trigger_verifica_e_remove_pessoas_e_referencias
BEFORE DELETE ON Filme
FOR EACH ROW
EXECUTE PROCEDURE verifica_e_remove_pessoas_e_referencias();
```

Exemplo 1, apaga pessoa: Vamos apagar o filme “Parasita”, exemplo com atores exclusivos desse filme no banco de dados atual

SQL - Função auxiliar (Retorna todos associados a um filme e a SOMENTE ele):

```
 CREATE OR REPLACE FUNCTION
lista_pessoas_exclusivas_por_filme(p_titulo_filme VARCHAR)
RETURNS TABLE (
    nome_pessoa VARCHAR,
    papel TEXT
) AS $$
BEGIN
    RETURN QUERY
    SELECT fa.Nome_Pessoa, 'Ator' AS Papel
    FROM Filme_Ator fa
    WHERE fa.Titulo_Filme = p_titulo_filme
    AND conta_obras_por_pessoa(fa.Nome_Pessoa) = 1

    UNION

    SELECT fd.Nome_Pessoa, 'Diretor' AS Papel
    FROM Filme_Diretor fd
    WHERE fd.Titulo_Filme = p_titulo_filme
    AND conta_obras_por_pessoa(fd.Nome_Pessoa) = 1

    UNION

    SELECT fp.Nome_Pessoa, 'Produtor' AS Papel
    FROM Filme_Produtor fp
    WHERE fp.Titulo_Filme = p_titulo_filme
    AND conta_obras_por_pessoa(fp.Nome_Pessoa) = 1;
END;
$$ LANGUAGE plpgsql;
```

SQL - Consultando filme “Parasita” com lista_pessoas_exclusivas_por_filme:

```
SELECT * FROM lista_pessoas_exclusivas_por_filme('Parasita');
```

nome_pessoa	papel
Song Kang-ho	Ator
Kwak Sin-ae	Produtor
Bong Joon-ho	Diretor
Park So-dam	Ator
Choi Woo-shik	Ator

SQL - Deletando filme:

```
DELETE FROM Filme WHERE Titulo = 'Parasita';
```

SQL - Pesquisando pessoas após deleção:

```
SELECT Nome_Pessoa
FROM Pessoa
WHERE Nome_Pessoa IN ('Song Kang-ho', 'Kwak Sin-ae', 'Bong Joon-ho', 'Park
So-dam', 'Choi Woo-shik');
```

Nenhuma linha encontrada.

Tempo de execução total: 2.679 ms

SQL executado.

Editar SQL

Exemplo 2, não apaga pessoa: Vamos apagar um filme “Interestelar” com “Christopher Nolan”, que aparece em mais de um filme.

Tabela diretor:

Ações		titulo_filme	nome_pessoa
Editar	Deletar	O Cavaleiro das Trevas	Christopher Nolan
Editar	Deletar	Interestelar	Christopher Nolan
Editar	Deletar	A Origem	Christopher Nolan

SQL - Usando “conta_obras_por_pessoa” com “Christopher Nolan”:

conta_obras_por_pessoa
3

Usando “lista_pessoas_exclusivas_por_filme” com “Interestelar”:

nome_pessoa	papel
Jessica Chastain	Ator
Matthew McConaughey	Ator
Anne Hathaway	Ator

SQL - Deletando filme:

```
DELETE FROM Filme WHERE Titulo = 'Interestelar';
```

SQL - Procurando pessoa (ainda está no banco de dados):

```
SELECT Nome_Pessoa
FROM Pessoa
WHERE Nome_Pessoa IN ('Christopher Nolan');
```

nome_pessoa
Christopher Nolan

SQL - Usando “conta_obras_por_pessoa” com “Christopher Nolan” (Agora com um filme a menos):

conta_obras_por_pessoa
2

Questão 13 (A)

Um exemplo claro de índice primário existente no banco de dados deste esquema relacional é o índice primário na tabela Filme. Este índice foi criado para garantir a unicidade do título de cada filme, facilitando a busca e recuperação rápida de informações associadas a filmes.

Declaração:

```
ALTER TABLE filmes.Filme ADD CONSTRAINT PK_Filme PRIMARY KEY (Titulo);
```

Justificativa do Índice Primário

Garantia de Unicidade

O índice primário assegura que cada filme tenha um título único, evitando duplicações e garantindo a integridade dos dados.

Melhoria no Desempenho das Consultas

Qualquer consulta que utilize o título do filme como critério (por exemplo, SELECT * FROM filmes.Filme WHERE Titulo = 'Um Sonho de Liberdade') se beneficia do índice primário, resultando em uma recuperação de dados mais eficiente.

Facilidade na Manutenção de Relacionamentos

O índice primário na tabela Filme é referenciado por várias outras tabelas através de chaves estrangeiras (e.g., Cotacao, Indicacao, Premiacao, Filme_Plataforma, Filme_Produtor, Filme_Diretor, Filme_Ator, e Filme_Genero). A existência do índice primário facilita a verificação e a manutenção da integridade referencial entre essas tabelas.

Utilização do Índice Primário

Consultas e Filtragens

O índice primário é constantemente utilizado em consultas que buscam filmes específicos pelo título. Isso resulta em uma recuperação mais rápida dos dados.

Join entre Tabelas

Quando tabelas relacionadas realizam junções (JOIN) com a tabela Filme, o índice primário permite um acesso mais rápido e eficiente aos registros correspondentes.

Manutenção da Integridade Referencial

Durante inserções, atualizações ou deleções, o índice primário assegura que as referências cruzadas entre tabelas sejam validadas rapidamente, mantendo a consistência e a integridade dos dados.

.

Questão 13 (B)

1) Índice em Nome_Pessoa da Tabela Pessoa

Justificativa: Este índice pode melhorar o desempenho das consultas que buscam informações sobre pessoas específicas (atores, diretores, produtores) pelo nome. Como as consultas frequentemente envolvem junções com outras tabelas para obter detalhes dos filmes relacionados às pessoas, um índice no campo Nome_Pessoa ajudará a acelerar essas operações.

```
CREATE INDEX idx_nome_pessoa ON filmes.Pessoa(Nome_Pessoa);
```

2) Índice em Ano_Lancamento da Tabela Filme

Justificativa: Este índice pode ser útil para consultas que filtram ou ordenam filmes por ano de lançamento. Consultas que buscam tendências históricas, ou que filtram filmes lançados em um determinado período, se beneficiarão desse índice.

```
CREATE INDEX idx_ano_lancamento ON filmes.Filme(Ano_Lancamento);
```

Testagem:

Primeiro:

```
EXPLAIN ANALYZE SELECT * FROM filmes.Pessoa WHERE Nome_Pessoa =
'Christopher Nolan';
```

```
```

QUERY PLAN
Seq Scan on pessoa (cost=0.00..5.58 rows=1 width=47)
Filter: ((nome_pessoa)::text = 'Christopher Nolan'::text)

Segundo:

```
EXPLAIN ANALYZE SELECT * FROM filmes.Filme WHERE Ano_Lancamento = 2020;
```

QUERY PLAN
Seq Scan on filme (cost=0.00..3.36 rows=1 width=1072)
Filter: (ano_lancamento = 2020)

Conclusão (Geral) - Índices Secundários:

Atualmente, devido à pequena quantidade de dados no banco (cerca de 40 filmes e 5 pessoas por filme), os índices criados (idx_nome_pessoa e idx_ano_lancamento) não estão sendo utilizados pelo otimizador de consultas. O otimizador determina que a varredura sequencial é mais eficiente com o volume de dados atual.

No entanto, os índices são potencialmente úteis. Se o banco de dados contiver milhões ou mesmo milhares de registros, os índices melhorariam significativamente o desempenho das consultas. Consultas que envolvem filtragem por nome de pessoa ou ano de lançamento se beneficiariam dos índices, resultando

em tempos de resposta mais rápidos e maior eficiência nas operações de consulta.

Portanto, embora os índices não sejam essenciais com a quantidade atual de dados, eles são uma preparação importante para o futuro crescimento do banco de dados e ajudarão a manter o desempenho à medida que o volume de dados aumentar.