```
-- Criar o banco de dados cinema


CREATE DATABASE IF NOT EXISTS cinema;
USE cinema;

-- Criar tabela cinema
CREATE TABLE IF NOT EXISTS cinema (
  id_cinema INT(11) NOT NULL AUTO_INCREMENT,
  nome VARCHAR(255) NOT NULL,
  endereco VARCHAR(255) NOT NULL,
  PRIMARY KEY (id_cinema)
) ENGINE = InnoDB;

-- Criar tabela cliente
CREATE TABLE IF NOT EXISTS cliente (
  cpf VARCHAR(11) NOT NULL,
  nome VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL,
  PRIMARY KEY (cpf)
) ENGINE = InnoDB;

-- Criar tabela filme
CREATE TABLE IF NOT EXISTS filme (
  id_filme INT(11) NOT NULL AUTO_INCREMENT,
  titulo VARCHAR(255) NOT NULL,
  diretor VARCHAR(255) NOT NULL,
  duracao TIME NOT NULL,
  PRIMARY KEY (id_filme)
) ENGINE = InnoDB;

-- Criar tabela sala
CREATE TABLE IF NOT EXISTS sala (
  id_sala INT(11) NOT NULL AUTO_INCREMENT,
  id_cinema INT(11) NOT NULL,
  numero INT(3) NOT NULL,
  capacidade INT(3) NOT NULL,
  PRIMARY KEY (id_sala),
  FOREIGN KEY (id_cinema) REFERENCES cinema(id_cinema) ON DELETE CASCADE
) ENGINE = InnoDB;

CREATE TABLE equipamento (
    id_equipamento INT PRIMARY KEY AUTO_INCREMENT,
    descricao VARCHAR(255) NOT NULL,
    tem_projetor_3d TINYINT NOT NULL
);

CREATE TABLE equipamento_has_sala (
```

```
  equipamento_id INT,
  sala_id INT,
  FOREIGN KEY (equipamento_id) REFERENCES equipamento(id_equipamento),
  FOREIGN KEY (sala_id) REFERENCES sala(id_sala)
);

-- Criar tabela sessao
CREATE TABLE IF NOT EXISTS sessao (
  id_sessao INT(11) NOT NULL AUTO_INCREMENT,
  id_filme INT(11) NOT NULL,
  id_sala INT(11) NOT NULL,
  horario DATETIME NOT NULL,
  PRIMARY KEY (id_sessao),
  FOREIGN KEY (id_filme) REFERENCES filme(id_filme) ON DELETE CASCADE,
  FOREIGN KEY (id_sala) REFERENCES sala(id_sala) ON DELETE CASCADE
) ENGINE = InnoDB;

-- Criar tabela ingresso
CREATE TABLE IF NOT EXISTS ingresso (
  id_ingresso INT(11) NOT NULL AUTO_INCREMENT,
  id_sessao INT(11) NOT NULL,
  cpf_cliente VARCHAR(11) NOT NULL,
  valor DECIMAL(5, 2) NOT NULL,
  PRIMARY KEY (id_ingresso),
  FOREIGN KEY (id_sessao) REFERENCES sessao(id_sessao) ON DELETE CASCADE,
  FOREIGN KEY (cpf_cliente) REFERENCES cliente(cpf) ON DELETE CASCADE
) ENGINE = InnoDB;

-- Criar tabela pagamento
CREATE TABLE IF NOT EXISTS pagamento (
  id_pagamento INT(11) NOT NULL AUTO_INCREMENT,
  id_ingresso INT(11) NOT NULL,
  tipo_pagamento VARCHAR(255) NOT NULL,
  PRIMARY KEY (id_pagamento),
  FOREIGN KEY (id_ingresso) REFERENCES ingresso(id_ingresso) ON DELETE
CASCADE
) ENGINE = InnoDB;

-- Criar tabela notafiscal
CREATE TABLE IF NOT EXISTS notafiscal (
  id_notafiscal INT(11) NOT NULL AUTO_INCREMENT,
  id_pagamento INT(11) NOT NULL,
  valor_total DECIMAL(5, 2) NOT NULL,
  PRIMARY KEY (id_notafiscal),
  FOREIGN KEY (id_pagamento) REFERENCES pagamento(id_pagamento) ON DELETE
CASCADE
) ENGINE = InnoDB;
```

```
-- Inserir dados nas tabelas
-- Tabela cinema
INSERT INTO cinema (nome, endereco) VALUES
  ('Cinema A', 'Endereço A'),
  ('Cinema B', 'Endereço B');

-- Tabela cliente
INSERT INTO cliente (cpf, nome, email) VALUES
  ('11111111111', 'Cliente A', 'clienteA@example.com'),
  ('22222222222', 'Cliente B', 'clienteB@example.com');

-- Tabela filme
INSERT INTO filme (titulo, diretor, duracao) VALUES
  ('Filme A', 'Diretor A', '02:30:00'),
  ('Filme B', 'Diretor B', '01:45:00');

-- Tabela sala
INSERT INTO sala (id_cinema, numero, capacidade) VALUES
  (1, 1, 100),
  (1, 2, 80),
  (2, 1, 120);

-- Tabela sessao
INSERT INTO sessao (id_filme, id_sala, horario) VALUES
  (1, 1, '2023-06-01 18:00:00'),
  (1, 1, '2023-06-01 20:30:00'),
  (2, 2, '2023-06-02 19:00:00');

-- Tabela ingresso
INSERT INTO ingresso (id_sessao, cpf_cliente, valor) VALUES
  (1, '11111111111', 25.00),
  (1, '22222222222', 25.00),
  (2, '11111111111', 25.00);

-- Tabela equipamento
INSERT INTO equipamento (descricao, tem_projetor_3d)
VALUES ('Projetor', 1);

-- Tabela pagamento
INSERT INTO pagamento (id_ingresso, tipo_pagamento) VALUES
  (1, 'Cartão de Crédito'),
  (2, 'Dinheiro'),
  (3, 'Cartão de Débito');

-- Tabela notafiscal
INSERT INTO notafiscal (id_pagamento, valor_total) VALUES
  (1, 25.00),
  (2, 25.00),
```

```
  (3, 25.00);

-- Cinema
-- UPDATE
UPDATE cinema SET endereco = 'Novo Endereço' WHERE id_cinema = 1;

-- Cliente
-- UPDATE
UPDATE cliente SET nome = 'Novo Nome' WHERE cpf = '12345678901';

-- Equipamento
-- UPDATE
UPDATE equipamento SET descricao = 'Nova Descrição' WHERE id_equipamento = 1;

-- Filme
-- UPDATE
UPDATE filme SET titulo = 'Novo Título' WHERE id_filme = 1;

-- Ingresso
-- UPDATE
UPDATE ingresso SET valor = 15.99 WHERE id_ingresso = 1;

-- SALA
-- UPDATE
UPDATE ingresso SET valor = 15.99 WHERE id_ingresso = 1;

--  PAGAMENTO
-- UPDATE
UPDATE pagamento SET tipo_pagamento = 'Cartão de Crédito' WHERE id_pagamento = 1;

-- notafiscal
-- UPDATE
UPDATE notafiscal SET valor_total = 50.99 WHERE id_notafiscal = 1;

-- B) SELECT com LIKE:

-- Neste caso, a consulta seleciona todas as colunas da tabela "filme"
-- onde o título contenha a letra "F". O operador "LIKE" permite fazer correspondências
-- parciais em strings, e o símbolo "%" é usado como um caractere curinga, representando
-- qualquer sequência de caracteres.
SELECT * FROM filme WHERE titulo LIKE '%F%';

-- C)SELECT com ORDER BY:

-- Nessa consulta, são selecionadas todas as colunas da tabela "cinema"
-- e os resultados são ordenados pelo campo "endereco". A cláusula "ORDER BY"
-- permite ordenar os resultados da consulta com base em uma ou mais colunas
especificadas.
```

```
SELECT * FROM cinema ORDER BY endereco;

-- D) SELECT com GROUP BY e HAVING:

-- Essa consulta seleciona a coluna "id_cinema" e calcula o número total de salas
-- para cada cinema na tabela "sala". A cláusula "GROUP BY" é usada para agrupar os
-- resultados com base no "id_cinema", e a cláusula "HAVING" filtra os grupos resultantes
-- para exibir apenas aqueles com mais de 0 salas.
SELECT id_cinema, COUNT(*) as total_salas
FROM sala
GROUP BY id_cinema
HAVING total_salas > 0;

-- E)  SELECT com JOIN com duas tabelas:

-- Nessa consulta, são selecionadas as colunas "id_pagamento" e "id_notafiscal" da tabela
-- "pagamento" e "tipo_pagamento" e "valor_total" da tabela "notafiscal". As tabelas são
-- unidas usando a cláusula "JOIN" e a condição de junção é que o "id_pagamento" da
tabela
-- "pagamento" seja igual ao "id_pagamento" da tabela "notafiscal".

SELECT p.id_pagamento, nf.id_notafiscal, p.tipo_pagamento, nf.valor_total
FROM pagamento p
JOIN notafiscal nf ON p.id_pagamento = nf.id_pagamento;

-- F) SELECT com JOIN com três tabelas:

-- Essa consulta seleciona as colunas "id_ingresso",
-- "horario" e "nome" da tabela "ingresso", "sessao" e "cliente", respectivamente.
-- As tabelas são unidas usando a cláusula "JOIN" e as condições de junção são que o
-- "id_sessao" da tabela "ingresso" seja igual ao "id_sessao" da tabela "sessao" e o
-- "cpf_cliente" da tabela "ingresso" seja igual ao "cpf" da tabela "cliente".

SELECT ingresso.id_ingresso, sessao.horario, cliente.nome
FROM ingresso
JOIN sessao ON ingresso.id_sessao = sessao.id_sessao
JOIN cliente ON ingresso.cpf_cliente = cliente.cpf;

-- G)SELECT com JOIN com quatro tabelas:

-- Nessa consulta, são selecionadas as colunas "id_notafiscal",
-- "tipo_pagamento", "nome" e "nome" da tabela "notafiscal", "pagamento",
-- "cliente" e "cinema", respectivamente. As tabelas são unidas usando a cláusula "JOIN"
-- e as condições de junção são especificadas entre as tabelas relacionadas.
SELECT notafiscal.id_notafiscal, pagamento.tipo_pagamento, cliente.nome, cinema.nome
FROM notafiscal
JOIN pagamento ON notafiscal.id_pagamento = pagamento.id_pagamento
JOIN ingresso ON pagamento.id_ingresso = ingresso.id_ingresso
```

```
JOIN cliente ON ingresso.cpf_cliente = cliente.cpf
JOIN sessao ON ingresso.id_sessao = sessao.id_sessao
JOIN sala ON sessao.id_sala = sala.id_sala
JOIN cinema ON sala.id_cinema = cinema.id_cinema;


-- H)SELECT com JOIN com no mínimo 3 tabelas, GROUP BY e HAVING:
-- Essa consulta seleciona as colunas "nome_cinema", "numero_sala"
-- e conta o número total de ingressos para cada combinação única de
-- "nome_cinema" e "numero_sala". As tabelas "cinema", "sala" e "sessao"
-- são unidas usando a cláusula "JOIN". Em seguida, os resultados são
-- agrupados usando a cláusula "GROUP BY" e, finalmente, a cláusula "HAVING"
-- filtra os grupos resultantes com base no número mínimo de ingressos (neste caso, 1 ou
mais).

SELECT c.nome AS nome_cinema, s.numero AS numero_sala, COUNT(*) AS
total_ingressos
FROM cinema c
JOIN sala s ON c.id_cinema = s.id_cinema
JOIN sessao se ON s.id_sala = se.id_sala
GROUP BY c.nome, s.numero
HAVING total_ingressos >= 1;

-- I)SELECT com JOIN com no mínimo 3 tabelas, GROUP BY (diferente do item h):

-- O resultado final será uma lista com o título do filme, o total de ingressos
-- vendidos para cada filme e o valor total das notas fiscais correspondentes a
-- esses ingressos. Cada linha representará uma combinação única de título do filme
-- e valor total da nota fiscal.

SELECT f.titulo, COUNT(*) AS total_ingressos, nf.valor_total AS total_valor
FROM ingresso i
JOIN sessao se ON i.id_sessao = se.id_sessao
JOIN filme f ON se.id_filme = f.id_filme
JOIN notafiscal nf ON i.id_ingresso = nf.id_pagamento
GROUP BY f.titulo, nf.valor_total;


-- J) SELECT com JOIN com no mínimo 2 tabelas, WHERE, GROUP BY e HAVING
(diferente do item h e i):

-- Nessa consulta, são selecionados o nome do cinema e o número total de filmes exibidos
em cada
-- cinema. As tabelas "cinema", "sala" e "sessao" são unidas usando a cláusula "JOIN". Os
-- resultados são agrupados por "nome" usando a cláusula "GROUP BY". A cláusula
"HAVING"
-- filtra os grupos resultantes com base no número mínimo de filmes (neste caso, 1 ou
mais).
```

-- A cláusula "WHERE" também pode ser usada para filtrar os dados antes do agrupamento.

```
SELECT c.nome, COUNT(*) AS total_filmes
FROM cinema c
JOIN sala s ON c.id_cinema = s.id_cinema
JOIN sessao se ON s.id_sala = se.id_sala
GROUP BY c.nome
HAVING total_filmes >=1;
```

```
-- Cinema
-- DELETE
DELETE FROM cinema WHERE id_cinema = 1;

-- Cliente
-- DELETE
DELETE FROM cliente WHERE cpf = '12345678901';


-- Equipamento
-- DELETE
DELETE FROM equipamento WHERE id_equipamento = 1;

-- Filme
-- DELETE
DELETE FROM filme WHERE id_filme = 1;

-- Ingresso
-- DELETE
DELETE FROM ingresso WHERE id_ingresso = 1;


-- SALA
-- DELETE
DELETE FROM ingresso WHERE id_ingresso = 1;

--  PAGAMENTO
-- DELETE
DELETE FROM pagamento WHERE id_pagamento = 1;

-- notafiscal
-- DELETE
DELETE FROM notafiscal WHERE id_notafiscal = 1;

-- Equipamento_has_sala
-- DELETE
```

```
DELETE FROM equipamento_has_sala WHERE equipamento_id = 1 AND sala_id = 1;
```