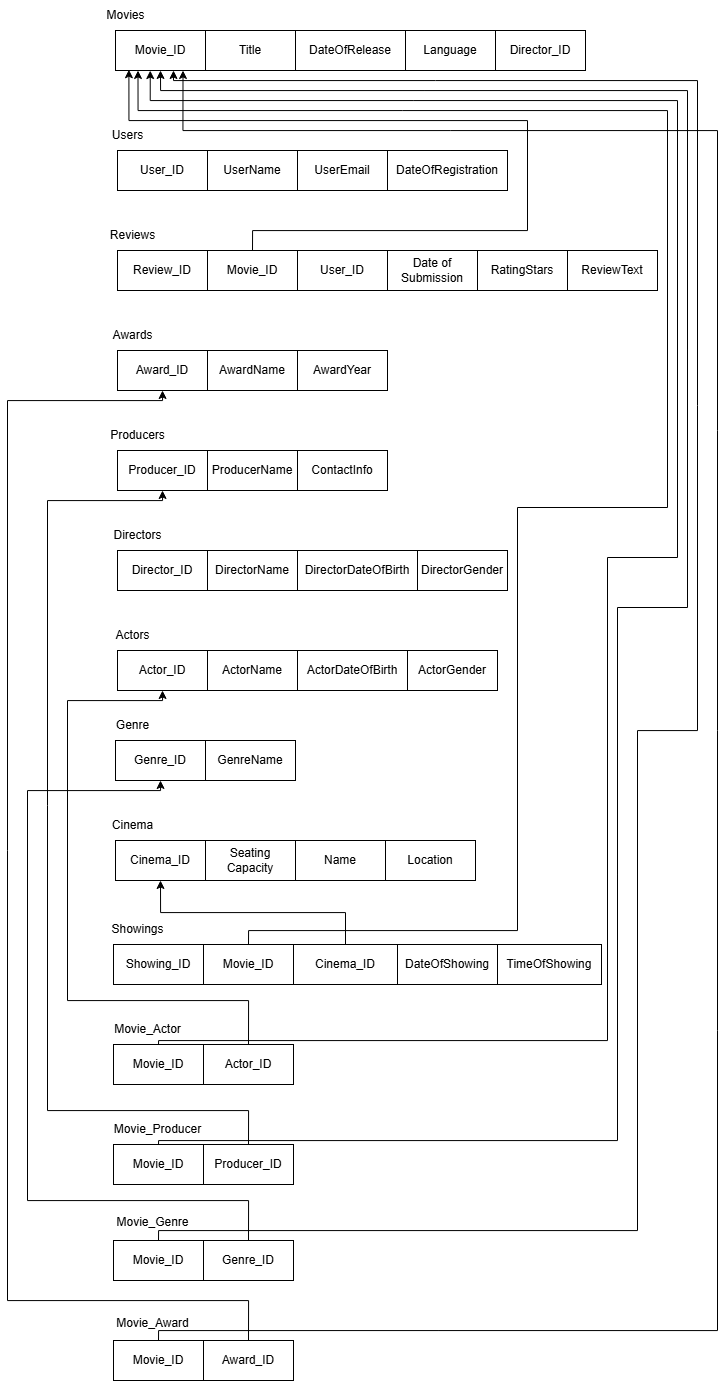
**RELATIONAL SCHEMA:**

****

**SETUP:**

sudo apt update

sudo apt install mysql-server

sudo mysql

**CODE FOR TASK 1:**

*-- Create movie\_db database*

*CREATE DATABASE movie\_db;*

*-- Use movie\_db database for operations*

*USE movie\_db;*

*-- Create Movies table*

*CREATE TABLE Movies (*

*movie\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*title VARCHAR(255) NOT NULL,*

*release\_date DATE,*

*language VARCHAR(50),*

*description TEXT*

*);*

*-- Create Actors table*

*CREATE TABLE Actors (*

*actor\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*name VARCHAR(255) NOT NULL,*

*date\_of\_birth DATE,*

*gender CHAR(1),*

*CHECK (gender IN ('M', 'F', 'O'))*

*);*

*-- Create Directors table*

*CREATE TABLE Directors (*

*director\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*name VARCHAR(255) NOT NULL,*

*date\_of\_birth DATE,*

*gender CHAR(1),*

*CHECK (gender IN ('M', 'F', 'O'))*

*);*

*-- Create Producers table*

*CREATE TABLE Producers (*

*producer\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*name VARCHAR(255) NOT NULL,*

*contact\_info VARCHAR(255)*

*);*

*-- Create Genres table*

*CREATE TABLE Genres (*

*genre\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*genre\_name VARCHAR(50) NOT NULL UNIQUE*

*);*

*-- Create Users table*

*CREATE TABLE Users (*

*user\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*name VARCHAR(255) NOT NULL,*

*email VARCHAR(255) NOT NULL UNIQUE,*

*registration\_date DATE DEFAULT (CURRENT\_DATE)*

*);*

*-- Create Reviews table*

*CREATE TABLE Reviews (*

*review\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*movie\_id INT,*

*user\_id INT,*

*rating INT NOT NULL,*

*review\_text TEXT,*

*submission\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,*

*FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) ON DELETE CASCADE,*

*FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,*

*CHECK (rating BETWEEN 1 AND 5)*

*);*

*-- Create Cinemas table*

*CREATE TABLE Cinemas (*

*cinema\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*name VARCHAR(255) NOT NULL,*

*location VARCHAR(255),*

*seating\_capacity INT,*

*CHECK (seating\_capacity > 0)*

*);*

*-- Create Showings table*

*CREATE TABLE Showings (*

*showing\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*movie\_id INT,*

*cinema\_id INT,*

*showing\_date DATE NOT NULL,*

*showing\_time TIME NOT NULL,*

*FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) ON DELETE CASCADE,*

*FOREIGN KEY (cinema\_id) REFERENCES Cinemas(cinema\_id) ON DELETE CASCADE*

*);*

*-- Create Awards table*

*CREATE TABLE Awards (*

*award\_id INT PRIMARY KEY AUTO\_INCREMENT,*

*award\_name VARCHAR(255) NOT NULL,*

*award\_year INT NOT NULL*

*);*

*-- Create Movie\_Actor junction table*

*CREATE TABLE Movie\_Actor (*

*movie\_id INT,*

*actor\_id INT,*

*PRIMARY KEY (movie\_id, actor\_id),*

*FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) ON DELETE CASCADE,*

*FOREIGN KEY (actor\_id) REFERENCES Actors(actor\_id) ON DELETE CASCADE*

*);*

*-- Create Movie\_Producer junction table*

*CREATE TABLE Movie\_Producer (*

*movie\_id INT,*

*producer\_id INT,*

*PRIMARY KEY (movie\_id, producer\_id),*

*FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) ON DELETE CASCADE,*

*FOREIGN KEY (producer\_id) REFERENCES Producers(producer\_id) ON DELETE CASCADE*

*);*

*-- Create Movie\_Genre junction table*

*CREATE TABLE Movie\_Genre (*

*movie\_id INT,*

*genre\_id INT,*

*PRIMARY KEY (movie\_id, genre\_id),*

*FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) ON DELETE CASCADE,*

*FOREIGN KEY (genre\_id) REFERENCES Genres(genre\_id) ON DELETE CASCADE*

*);*

*-- Create Movie\_Award junction table*

*CREATE TABLE Movie\_Award (*

*movie\_id INT,*

*award\_id INT,*

*PRIMARY KEY (movie\_id, award\_id),*

*FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) ON DELETE CASCADE,*

*FOREIGN KEY (award\_id) REFERENCES Awards(award\_id) ON DELETE CASCADE*

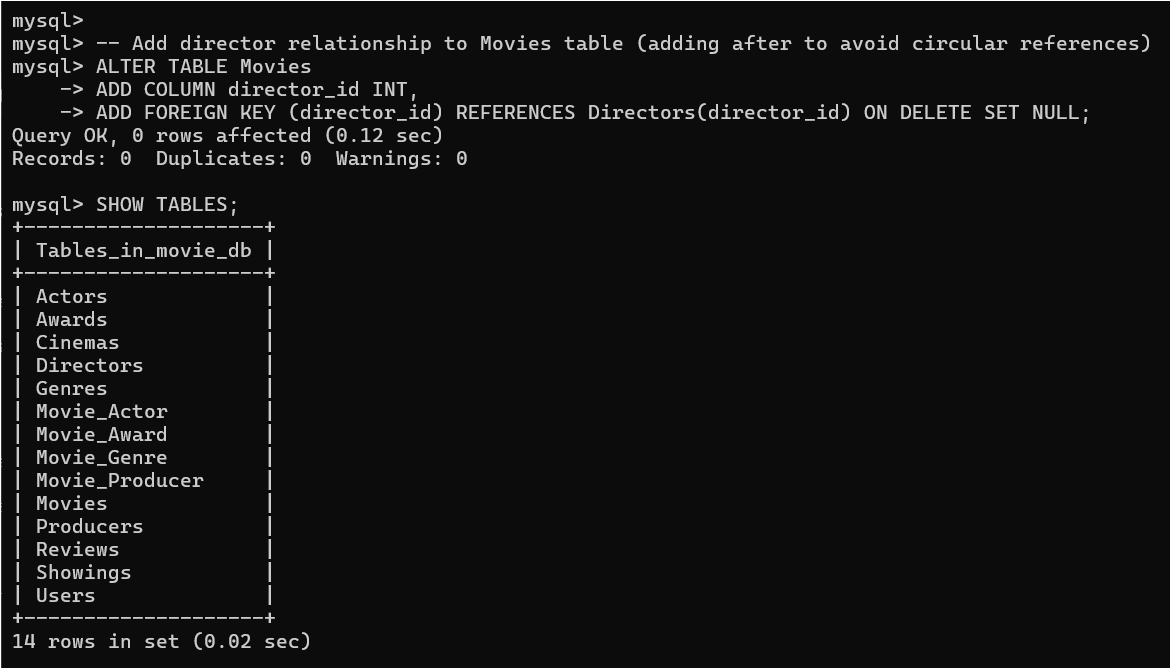
*);*

*-- Add director relationship to Movies table (adding after to avoid circular references)*

*ALTER TABLE Movies*

*ADD COLUMN director\_id INT,*

*ADD FOREIGN KEY (director\_id) REFERENCES Directors(director\_id) ON DELETE SET NULL;*



**CODE FOR TASK 2:**

**1. Valid Data Insertions:**

*-- Insert Directors*

*INSERT INTO Directors (name, date\_of\_birth, gender) VALUES*

*('Christopher Nolan', '1970-07-30', 'M'),*

*('Greta Gerwig', '1983-08-04', 'F'),*

*('Martin Scorsese', '1942-11-17', 'M');*

*-- Insert Movies*

*INSERT INTO Movies (title, release\_date, language, description, director\_id) VALUES*

*('Inception', '2010-07-16', 'English', 'A thief who enters dreams', 1),*

*('Barbie', '2023-07-21', 'English', 'Adventure of Barbie', 2),*

*('Killers of the Flower Moon', '2023-10-20', 'English', 'Native American murders investigation', 3);*

*-- Insert Actors*

*INSERT INTO Actors (name, date\_of\_birth, gender) VALUES*

*('Leonardo DiCaprio', '1974-11-11', 'M'),*

*('Margot Robbie', '1990-07-02', 'F'),*

*('Robert De Niro', '1943-08-17', 'M');*

*-- Insert Movie\_Actor relationships*

*INSERT INTO Movie\_Actor (movie\_id, actor\_id) VALUES*

*(1, 1), -- Inception - DiCaprio*

*(2, 2), -- Barbie - Robbie*

*(3, 1), -- Killers - DiCaprio*

*(3, 3); -- Killers - De Niro*

*-- Insert Producers*

*INSERT INTO Producers (name, contact\_info) VALUES*

*('Emma Thomas', 'emma.thomas@syncopy.com'),*

*('Tom Ackerley', 'tom.ackerley@luckychap.com'),*

*('Dan Friedkin', 'dan.friedkin@imperative.com');*

*-- Insert Movie\_Producer relationships*

*INSERT INTO Movie\_Producer (movie\_id, producer\_id) VALUES*

*(1, 1),*

*(2, 2),*

*(3, 3);*

*-- Insert Genres*

*INSERT INTO Genres (genre\_name) VALUES*

*('Action'),*

*('Comedy'),*

*('Drama');*

*-- Insert Movie\_Genre relationships*

*INSERT INTO Movie\_Genre (movie\_id, genre\_id) VALUES*

*(1, 1), -- Inception - Action*

*(2, 2), -- Barbie - Comedy*

*(3, 3); -- Killers - Drama*

*-- Insert Users*

*INSERT INTO Users (name, email) VALUES*

*('John Doe', 'john.doe@email.com'),*

*('Jane Smith', 'jane.smith@email.com'),*

*('Bob Wilson', 'bob.wilson@email.com');*

*-- Insert Reviews*

*INSERT INTO Reviews (movie\_id, user\_id, rating, review\_text) VALUES*

*(1, 1, 5, 'Masterpiece of cinema'),*

*(2, 2, 4, 'Surprisingly deep and entertaining'),*

*(3, 3, 5, 'Epic historical drama');*

*-- Insert Cinemas*

*INSERT INTO Cinemas (name, location, seating\_capacity) VALUES*

*('AMC Empire', 'New York', 500),*

*('Regal LA Live', 'Los Angeles', 400),*

*('Alamo Drafthouse', 'Austin', 200);*

*-- Insert Showings*

*INSERT INTO Showings (movie\_id, cinema\_id, showing\_date, showing\_time) VALUES*

*(1, 1, '2024-02-05', '18:00:00'),*

*(2, 2, '2024-02-05', '19:00:00'),*

*(3, 3, '2024-02-05', '20:00:00');*

*-- Insert Awards*

*INSERT INTO Awards (award\_name, award\_year) VALUES*

*('Best Picture Oscar', 2023),*

*('Best Director Golden Globe', 2023),*

*('Best Actor BAFTA', 2023);*

*-- Insert Movie\_Award relationships*

*INSERT INTO Movie\_Award (movie\_id, award\_id) VALUES*

*(1, 1),*

*(2, 2),*

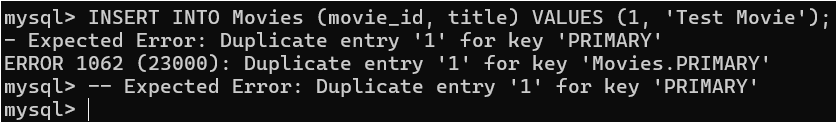
*(3, 3);*

**2. Invalid Data Insertions (These will generate errors):**

*-- Test Case 1: Duplicate Primary Key*

*INSERT INTO Movies (movie\_id, title) VALUES (1, 'Test Movie');*

*-- Expected Error: Duplicate entry '1' for key 'PRIMARY'*

**

*-- Test Case 2: Null Primary Key*

*INSERT INTO Movies (movie\_id, title) VALUES (NULL, 'Test Movie');*

*-- Expected Error: Column 'movie\_id' cannot be null*

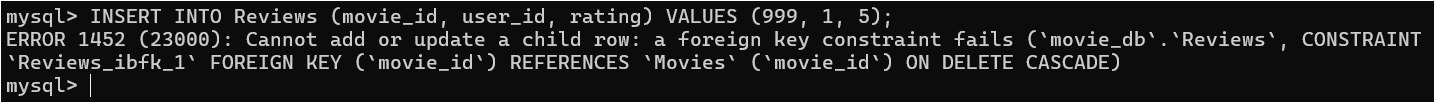
**

In reality, there’s no error, but it is expected to throw an error.

*-- Test Case 3: Foreign Key Violation*

*INSERT INTO Reviews (movie\_id, user\_id, rating) VALUES (999, 1, 5);*

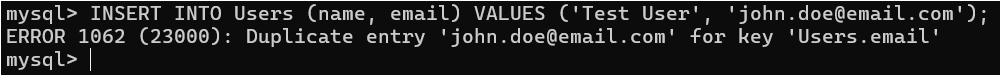
*-- Expected Error: Cannot add or update a child row: a foreign key constraint fails*

**

*-- Test Case 4: Duplicate Unique Value*

*INSERT INTO Users (name, email) VALUES ('Test User', 'john.doe@email.com');*

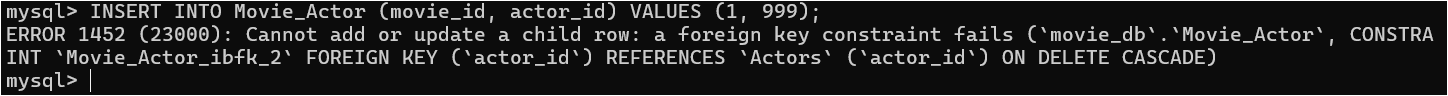
*-- Expected Error: Duplicate entry 'john.doe@email.com' for key 'email'*

**

*-- Test Case 5: Invalid Foreign Key Reference*

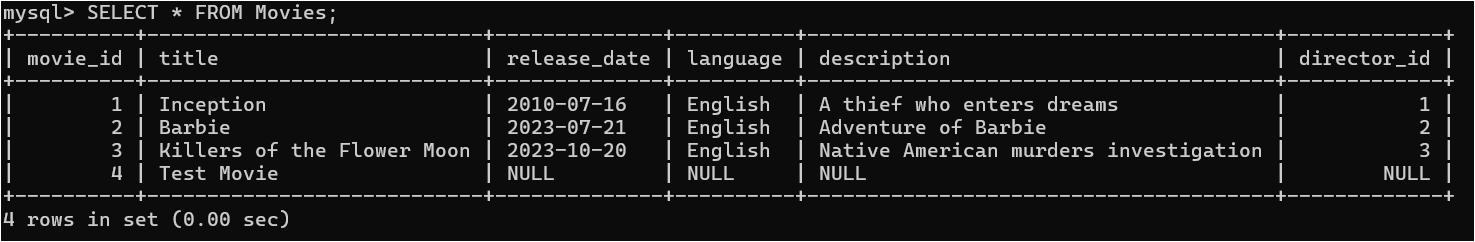
*INSERT INTO Movie\_Actor (movie\_id, actor\_id) VALUES (1, 999);*

*-- Expected Error: Cannot add or update a child row: a foreign key constraint fails*

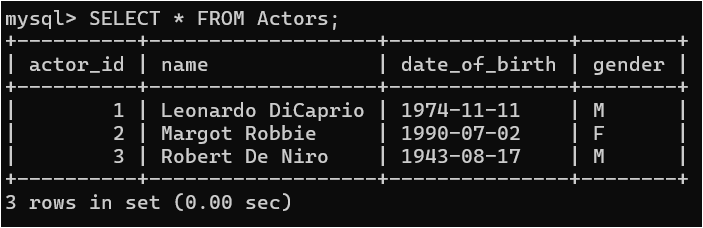
**

**3. View inserted data:**

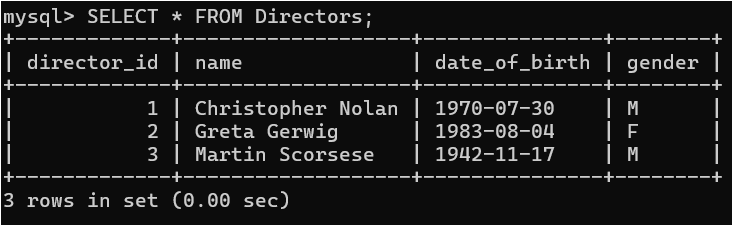
*SELECT \* FROM Movies;*

**

*SELECT \* FROM Actors;*

**

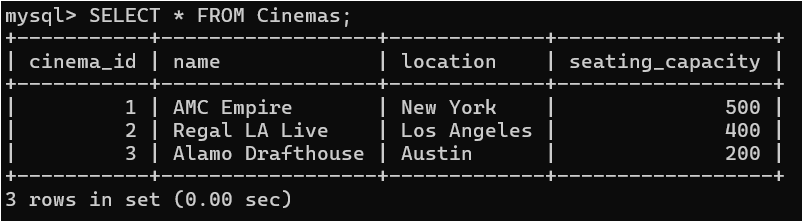
*SELECT \* FROM Directors;*

**

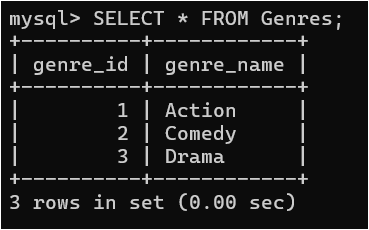
*SELECT \* FROM Awards;*

**

*SELECT \* FROM Cinemas;*

**

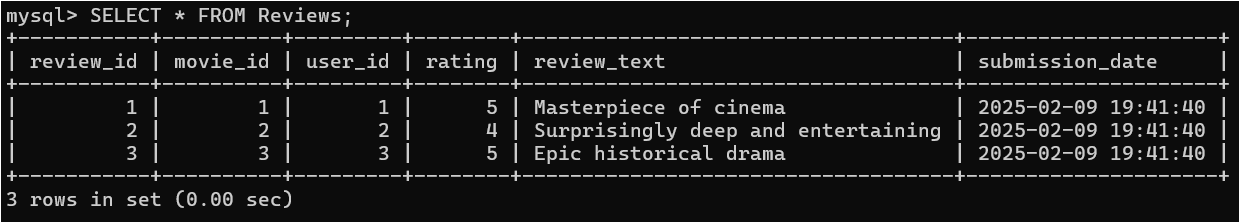
*SELECT \* FROM Genres;*

**

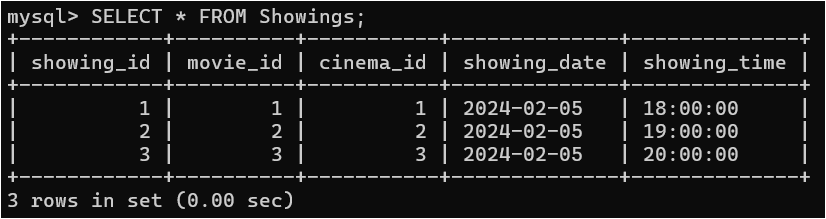
*SELECT \* FROM Producers;*

**

*SELECT \* FROM Reviews;*

**

*SELECT \* FROM Showings;*

**

*SELECT \* FROM Users;*

**

**Observations and Results**

**1. Valid Data Insertions:**

* Successfully inserted 3+ records in each table
* Created proper relationships between tables using junction table
* Default values for dates and timestamps work correctly

**2. Constraint Violations and Error Cases:**

a) Primary Key Constraints:

* Attempting to insert duplicate movie\_id fails
* Attempting to insert *NULL* primary key fails
* Auto-increment works correctly for new insertions

b) Foreign Key Constraints:

* Cannot insert reviews for non-existent movies
* Cannot insert movie-actor relationships for non-existent actors
* *ON DELETE CASCADE* works properly (can be tested by deleting a movie)

c) Unique Constraints:

* Cannot insert duplicate email addresses in Users table
* Cannot insert duplicate genre names in Genres table

**3. Data Integrity Observations:**

* The database maintains referential integrity
* Junction tables properly handle many-to-many relationships
* Default values are properly assigned when not specified

**To test these insertions:**

1. Run the valid insertions first:

*-- Copy and paste the valid insertions section*

2. Then try each error case separately to observe the error messages:

*-- Run each invalid insertion case one by one*

3. To verify the data:

*-- View inserted data*

*SELECT \* FROM Movies;*

*SELECT \* FROM Actors;*

*SELECT \* FROM Directors;*

*-- etc.*

**CODE FOR TASK 3:**

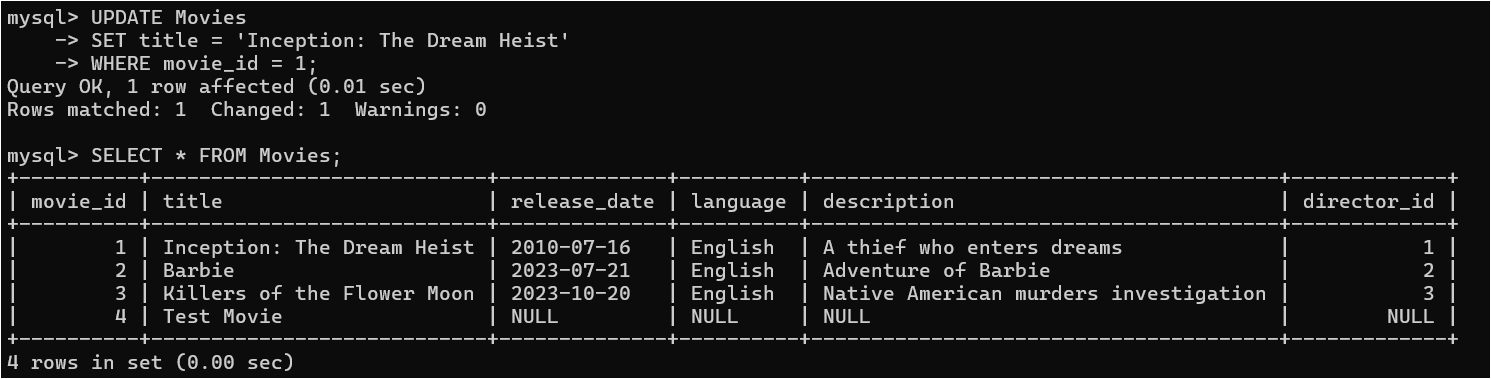
**1. UPDATE OPERATIONS:**

*-- Update 1: Simple update of a movie title*

*UPDATE Movies*

*SET title = 'Inception: The Dream Heist'*

*WHERE movie\_id = 1;*

**

*-- Update 2: Try to update a director\_id to non-existent value*

*-- This will fail due to foreign key constraint*

*UPDATE Movies*

*SET director\_id = 999*

*WHERE movie\_id = 1;*

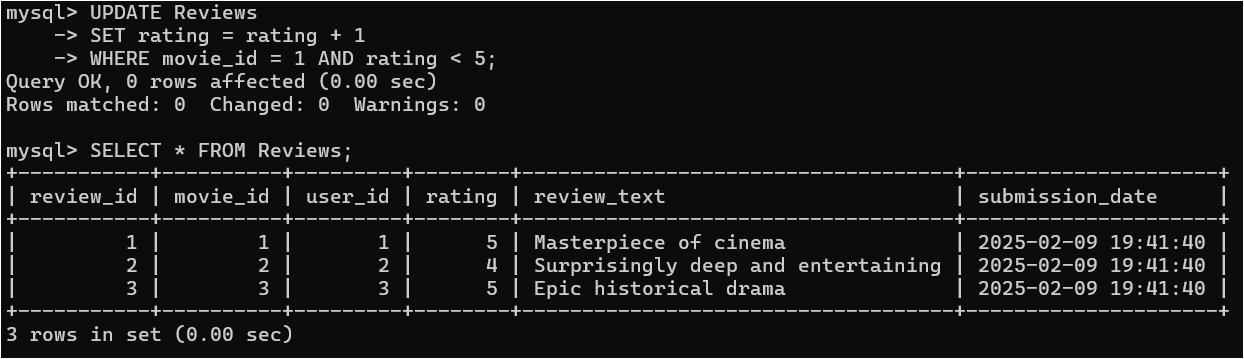
**

*-- Update 3: Update multiple reviews for a movie*

*UPDATE Reviews*

*SET rating = rating + 1*

*WHERE movie\_id = 1 AND rating < 5;*

**

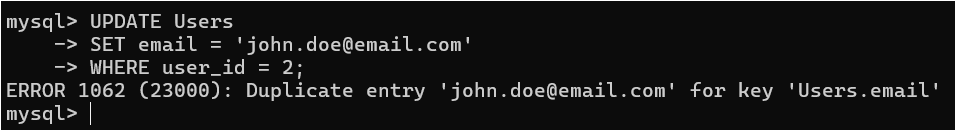
*-- Update 4: Try to update email to duplicate value*

*-- This will fail due to unique constraint*

*UPDATE Users*

*SET email = 'john.doe@email.com'*

*WHERE user\_id = 2;*

**

*-- Update 5: Update movie release date*

*UPDATE Movies*

*SET release\_date = '2024-01-01'*

*WHERE movie\_id = 1;*

**

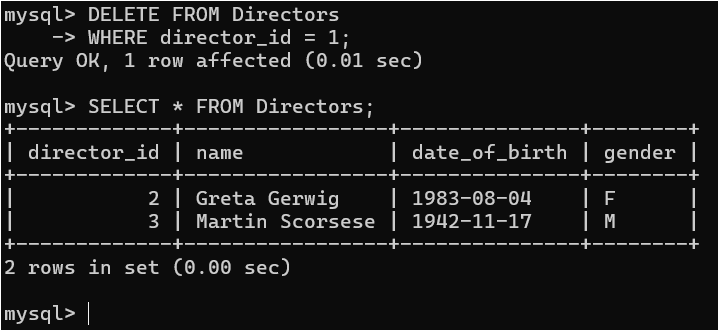
**2. DELETE OPERATIONS:**

*-- Delete 1: Try to delete a director who has movies*

*-- This will fail due to foreign key constraint unless we set ON DELETE CASCADE*

*DELETE FROM Directors*

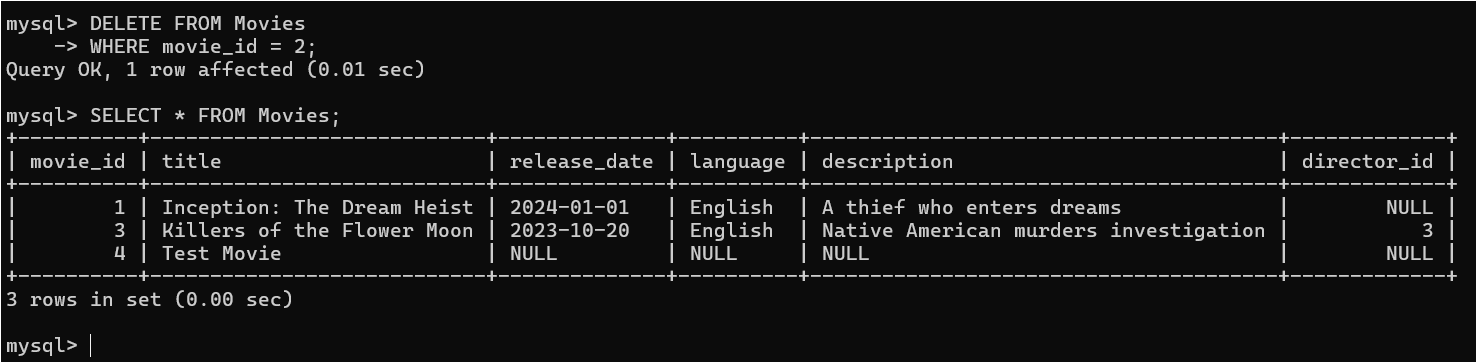
*WHERE director\_id = 1;*



*-- Delete 2: Delete a movie (this will cascade)*

*DELETE FROM Movies*

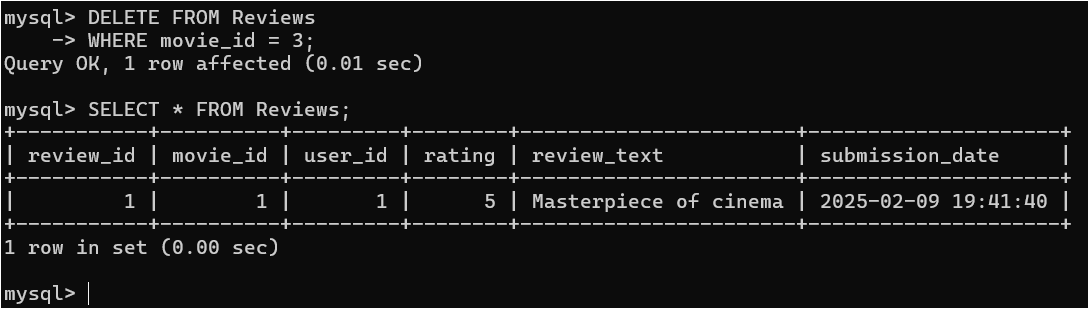
*WHERE movie\_id = 2;*

**

*-- Delete 3: Delete all reviews for a specific movie*

*DELETE FROM Reviews*

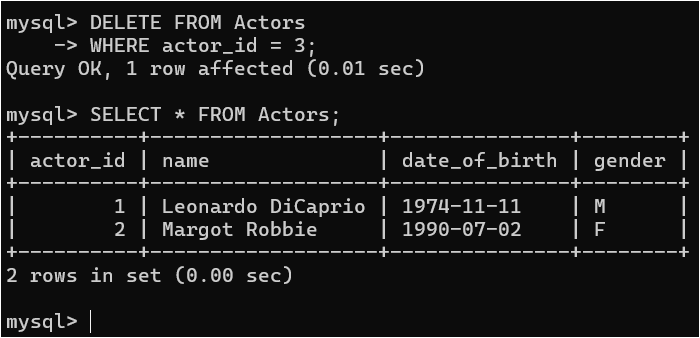
*WHERE movie\_id = 3;*

**

*-- Delete 4: Delete an actor (this will affect Movie\_Actor junction table due to CASCADE)*

*DELETE FROM Actors*

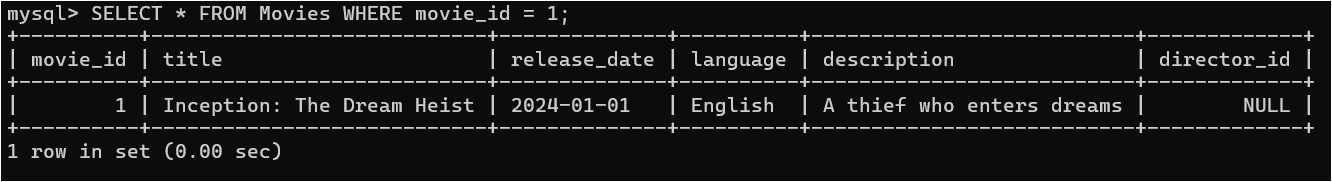
*WHERE actor\_id = 3;*

**

**3. VERIFICATION QUERIES:**

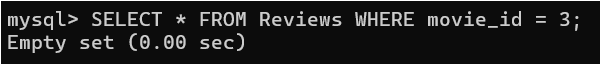
*-- Check updated movie title*

*SELECT \* FROM Movies WHERE movie\_id = 1;*

**

*-- Check if reviews were deleted (after Delete 3)*

*SELECT \* FROM Reviews WHERE movie\_id = 3;*

**

*-- Check if Movie\_Actor entries were deleted (after Delete 4)*

*SELECT \* FROM Movie\_Actor WHERE actor\_id = 3;*

**

*-- Check if movie deletion cascaded to related tables*

*SELECT \* FROM Reviews WHERE movie\_id = 2;*

**

*SELECT \* FROM Movie\_Actor WHERE movie\_id = 2;*

**

*SELECT \* FROM Movie\_Genre WHERE movie\_id = 2;*

**

**Explanation of Constraint Violations and Results**

**1. Foreign Key Constraints:**

*-- This will fail*

*UPDATE Movies SET director\_id = 999 WHERE movie\_id = 1;*

Error: Cannot add or update a child row: foreign key constraint fails

Explanation: You cannot reference a director\_id that doesn't exist in the Directors table

**2. Unique Constraints:**

*-- This will fail*

*UPDATE Users SET email = 'john.doe@email.com' WHERE user\_id = 2;*

Error: Duplicate entry 'john.doe@email.com' for key 'email'

Explanation: Email must be unique across all users

**3. Cascading Deletes:**

***--*** *This will fail*

*DELETE FROM Movies WHERE movie\_id = 2;*

Result: Successful deletion of the movie and all related records

Effects:

* Deletes all reviews for the movie
* Deletes all movie-actor relationships
* Deletes all movie-genre relationships
* Deletes all showings for the movie

**4. Reference Constraints:**

*-- This will fail unless all referenced movies are deleted first*

*DELETE FROM Directors WHERE director\_id = 1;*

Error: Cannot delete or update a parent row: a foreign key constraint fails

Explanation: Cannot delete a director who still has movies referencing them

**To test all these operations:**

1. First, run verification queries to check current state:

*SELECT \* FROM Movies;*

*SELECT \* FROM Directors;*

*SELECT \* FROM Reviews;*

2. Try the update operations and observe results:

*-- Run each UPDATE statement and check results*

3. Try the delete operations and observe cascading effects:

*-- Run each DELETE statement and check affected tables*

4. Final verification:

*-- Run all verification queries to see final state*

**Important Points to Note:**

1. *ON DELETE CASCADE* ensures referential integrity by automatically deleting related records

2. Foreign key constraints prevent orphaned records

3. Unique constraints ensure data consistency

4. Primary key constraints maintain record uniqueness

5. The database maintains integrity even during complex operations