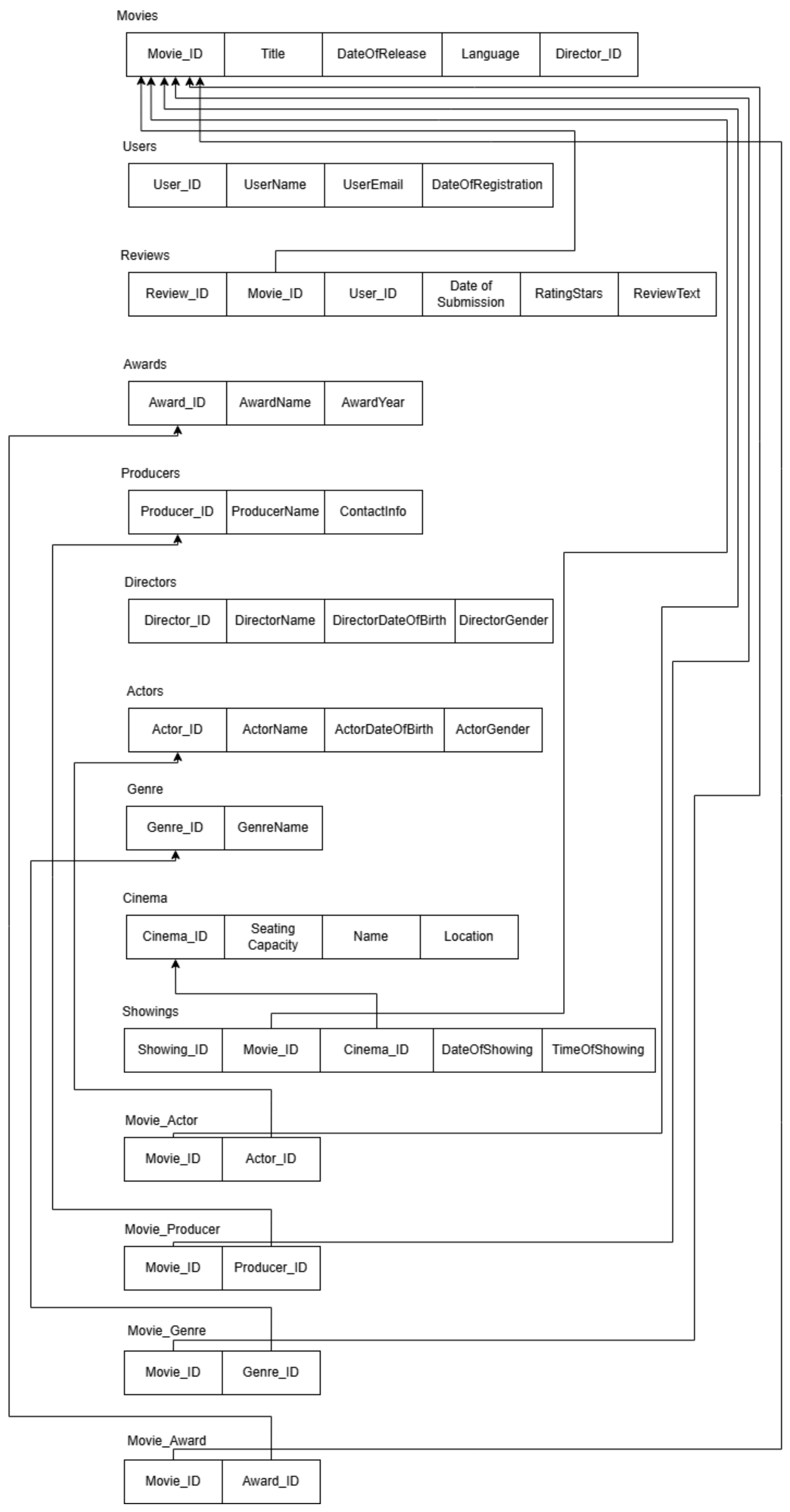
RELATIONAL SCHEMA:


Movies
Movie_ID | Title | DateOfRelease | Language | Director_ID
Users
User_ID | UserName | UserEmail | DateOfRegistration
Reviews
Review_ID | Movie_ID | User_ID | Date of Submission | RatingStars | ReviewText
Awards
Award_ID | AwardName | AwardYear
Producers
Producer_ID | ProducerName | ContactInfo
Directors
Director_ID | DirectorName | DirectorDateOfBirth | DirectorGender
Actors
Actor_ID | ActorName | ActorDateOfBirth | ActorGender
Genre
Genre_ID | GenreName
Cinema
Cinema_ID | Seating Capacity | Name | Location
Showings
Showing_ID | Movie_ID | Cinema_ID | DateOfShowing | TimeOfShowing
Movie_Actor
Movie_ID | Actor_ID
Movie_Producer
Movie_ID | Producer_ID
Movie_Genre
Movie_ID | Genre_ID
Movie_Award
Movie_ID | Award_ID

SETUP:

```
sudo apt update
sudo apt install mysql-server

sudo mysql
```

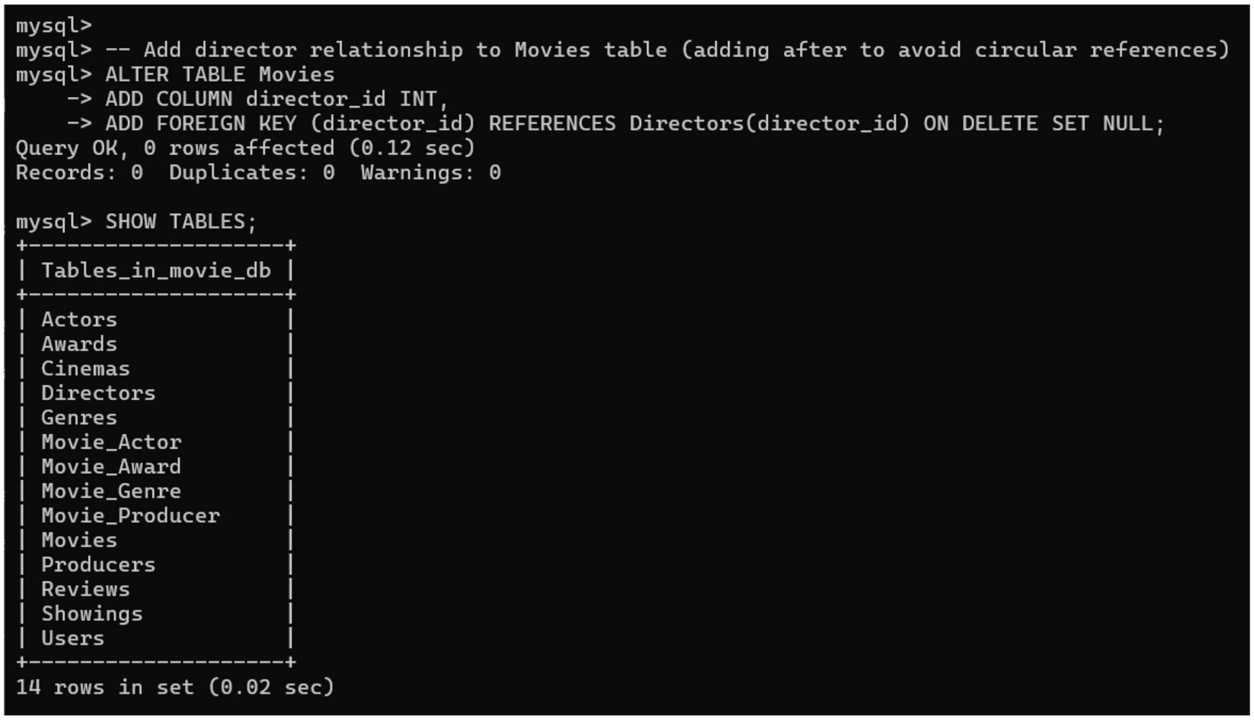
CODE FOR TASK 1:

```
-- 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;
```



```
mysql>
mysql> -- Add director relationship to Movies table (adding after to avoid circular references)
mysql> ALTER TABLE Movies
    -> ADD COLUMN director_id INT,
    -> ADD FOREIGN KEY (director_id) REFERENCES Directors(director_id) ON DELETE SET NULL;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW TABLES;
+--------------------+
| Tables_in_movie_db |
+--------------------+
| Actors             |
| Awards             |
| Cinemas            |
| Directors          |
| Genres             |
| Movie_Actor        |
| Movie_Award        |
| Movie_Genre        |
| Movie_Producer     |
| Movies             |
| Producers          |
| Reviews            |
| Showings           |
| Users              |
+--------------------+
14 rows in set (0.02 sec)
```

CODE FOR TASK 2:

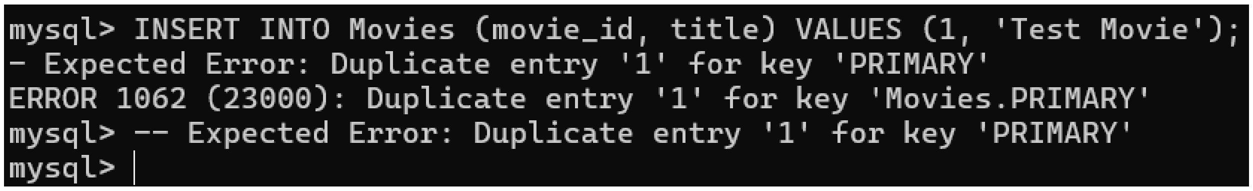
1. Valid Data Insertions:

```
-- 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);
```

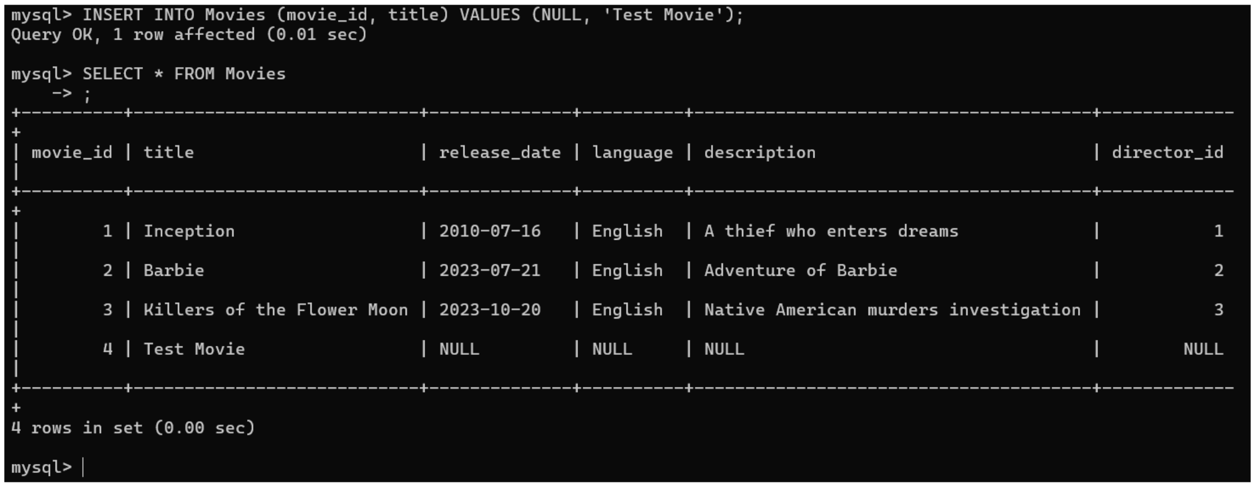
2. Invalid Data Insertions (These will generate errors):

-- Test Case 1: Duplicate Primary Key
INSERT INTO Movies (movie_id, title) VALUES (1, 'Test Movie');
-- Expected Error: Duplicate entry '1' for key 'PRIMARY'



```
mysql> INSERT INTO Movies (movie_id, title) VALUES (1, 'Test Movie');
- Expected Error: Duplicate entry '1' for key 'PRIMARY'
ERROR 1062 (23000): Duplicate entry '1' for key 'Movies.PRIMARY'
mysql> -- Expected Error: Duplicate entry '1' for key 'PRIMARY'
mysql> |
```

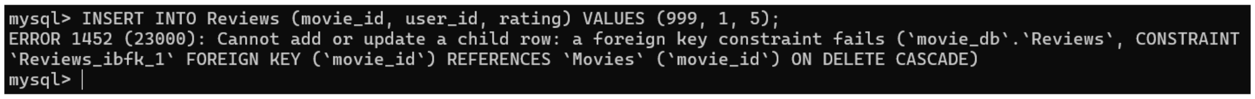
-- Test Case 2: Null Primary Key
INSERT INTO Movies (movie_id, title) VALUES (NULL, 'Test Movie');
-- Expected Error: Column 'movie_id' cannot be null



```
mysql> INSERT INTO Movies (movie_id, title) VALUES (NULL, 'Test Movie');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Movies
    -> ;
+----------+----------------------------+--------------+----------+----------------------------------------+-------------
+
| movie_id | title                      | release_date | language | description                            | director_id
|
+----------+----------------------------+--------------+----------+----------------------------------------+-------------
+
|        1 | Inception                  | 2010-07-16   | English  | A thief who enters dreams              |           1
|
|        2 | Barbie                     | 2023-07-21   | English  | Adventure of Barbie                    |           2
|
|        3 | Killers of the Flower Moon | 2023-10-20   | English  | Native American murders investigation |           3
|
|        4 | Test Movie                 | NULL         | NULL     | NULL                                   |        NULL
|
+----------+----------------------------+--------------+----------+----------------------------------------+-------------
+
4 rows in set (0.00 sec)

mysql> |
```

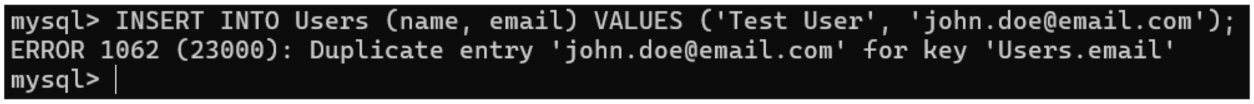
In reality, there’s no error, but it is expected to throw an error.

-- 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



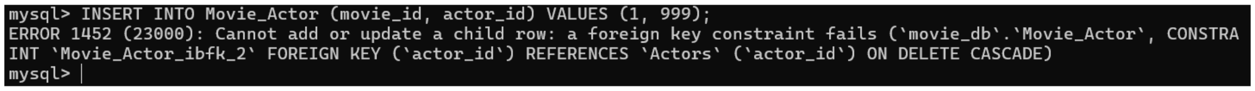
```
mysql> INSERT INTO Reviews (movie_id, user_id, rating) VALUES (999, 1, 5);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`movie_db`.`Reviews`, CONSTRAINT
`Reviews_ibfk_1` FOREIGN KEY (`movie_id`) REFERENCES `Movies` (`movie_id`) ON DELETE CASCADE)
mysql> |
```

-- 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'



```
mysql> INSERT INTO Users (name, email) VALUES ('Test User', 'john.doe@email.com');
ERROR 1062 (23000): Duplicate entry 'john.doe@email.com' for key 'Users.email'
mysql> |
```

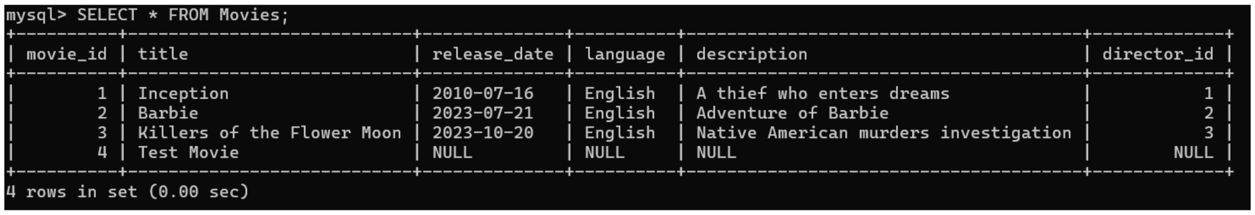
-- Test Case 5: Invalid Foreign Key Reference
INSERT INTO Movie_Actor (movie_id, actor_id) VALUES (1, 999);
-- Expected Error: Cannot add or update a child row: a foreign key constraint fails



```
mysql> INSERT INTO Movie_Actor (movie_id, actor_id) VALUES (1, 999);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`movie_db`.`Movie_Actor`, CONSTRA
INT `Movie_Actor_ibfk_2` FOREIGN KEY (`actor_id`) REFERENCES `Actors` (`actor_id`) ON DELETE CASCADE)
mysql> |
```

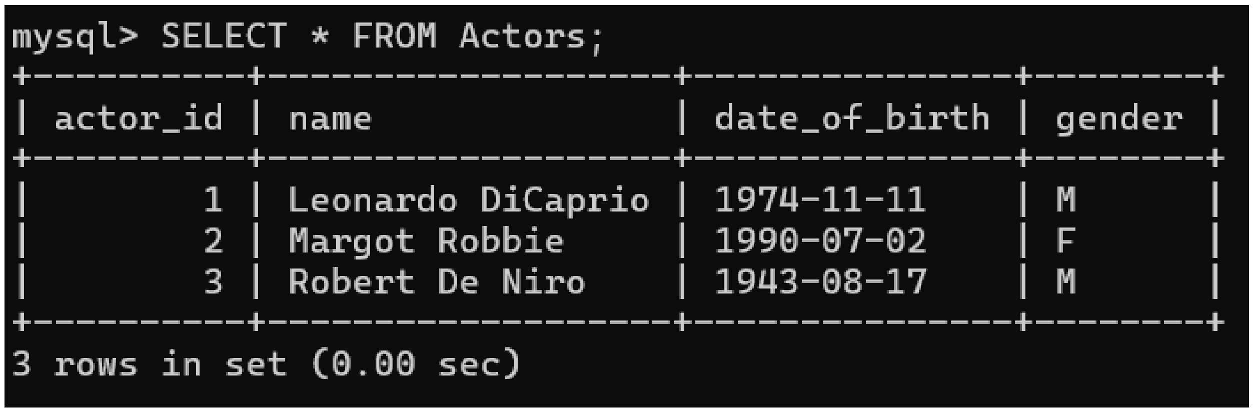
3. View inserted data:

*SELECT * FROM Movies;*



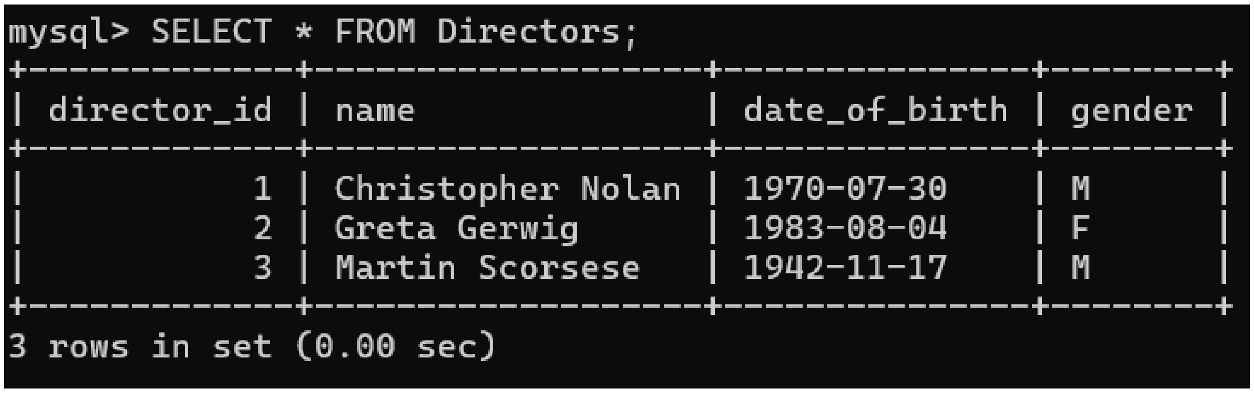
```
mysql> SELECT * FROM Movies;
+----------+---------------------------+--------------+----------+----------------------------------------+-------------+
| movie_id | title                     | release_date | language | description                            | director_id |
+----------+---------------------------+--------------+----------+----------------------------------------+-------------+
|        1 | Inception                 | 2010-07-16   | English  | A thief who enters dreams              |           1 |
|        2 | Barbie                    | 2023-07-21   | English  | Adventure of Barbie                    |           2 |
|        3 | Killers of the Flower Moon | 2023-10-20  | English  | Native American murders investigation  |           3 |
|        4 | Test Movie                | NULL         | NULL     | NULL                                   |        NULL |
+----------+---------------------------+--------------+----------+----------------------------------------+-------------+
4 rows in set (0.00 sec)
```

*SELECT * FROM Actors;*



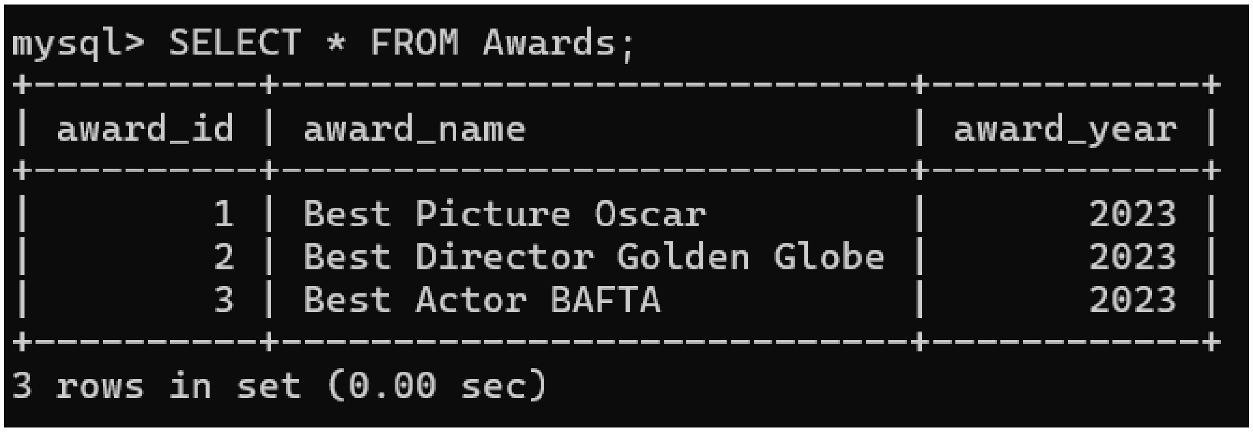
```
mysql> SELECT * FROM Actors;
+----------+-------------------+---------------+--------+
| actor_id | name              | date_of_birth | gender |
+----------+-------------------+---------------+--------+
|        1 | Leonardo DiCaprio | 1974-11-11    | M      |
|        2 | Margot Robbie     | 1990-07-02    | F      |
|        3 | Robert De Niro    | 1943-08-17    | M      |
+----------+-------------------+---------------+--------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Directors;*



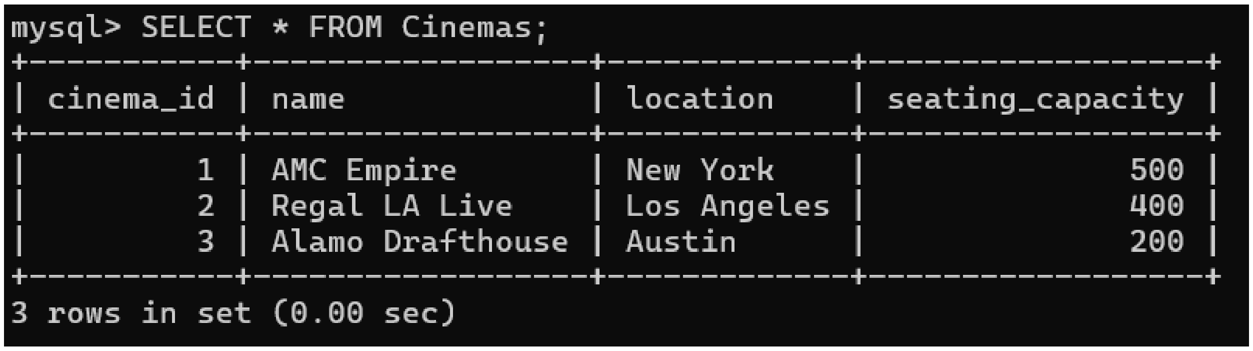
```
mysql> SELECT * FROM Directors;
+-------------+--------------------+---------------+--------+
| director_id | name               | date_of_birth | gender |
+-------------+--------------------+---------------+--------+
|           1 | Christopher Nolan  | 1970-07-30    | M      |
|           2 | Greta Gerwig       | 1983-08-04    | F      |
|           3 | Martin Scorsese    | 1942-11-17    | M      |
+-------------+--------------------+---------------+--------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Awards;*



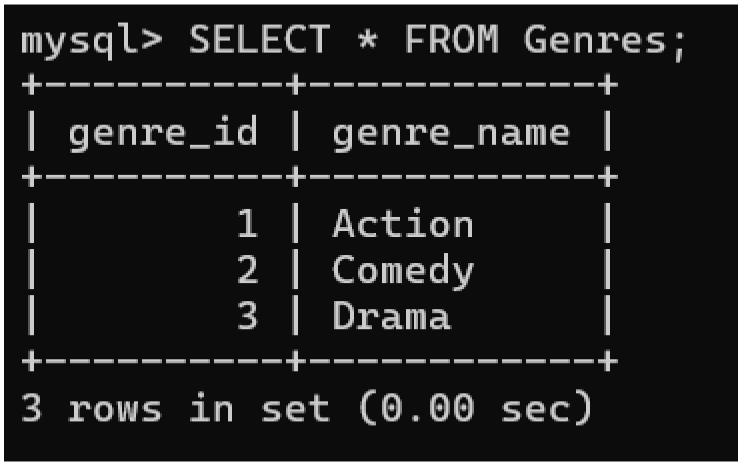
```
mysql> SELECT * FROM Awards;
+----------+-----------------------------+------------+
| award_id | award_name                  | award_year |
+----------+-----------------------------+------------+
|        1 | Best Picture Oscar          |       2023 |
|        2 | Best Director Golden Globe  |       2023 |
|        3 | Best Actor BAFTA            |       2023 |
+----------+-----------------------------+------------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Cinemas;*



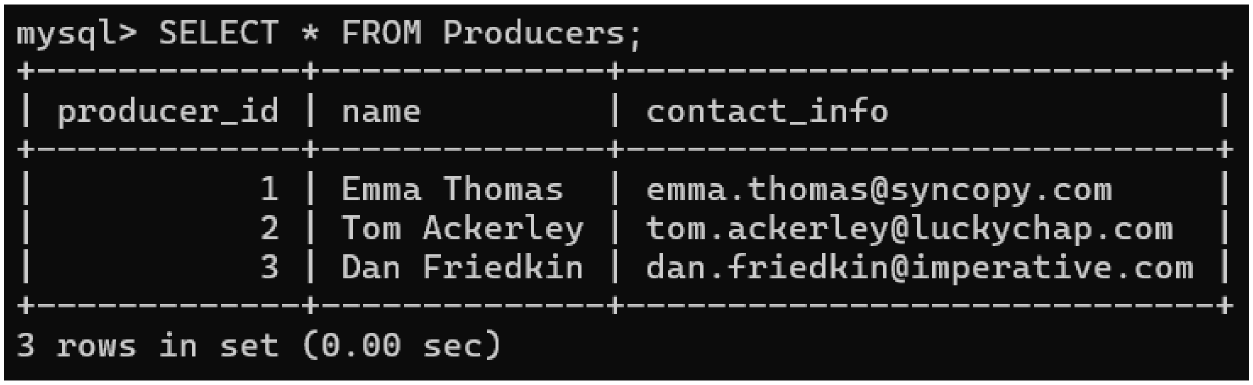
```
mysql> SELECT * FROM Cinemas;
+-----------+-----------------+-------------+------------------+
| cinema_id | name            | location    | seating_capacity |
+-----------+-----------------+-------------+------------------+
|         1 | AMC Empire      | New York    |              500 |
|         2 | Regal LA Live   | Los Angeles |              400 |
|         3 | Alamo Drafthouse| Austin      |              200 |
+-----------+-----------------+-------------+------------------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Genres;*



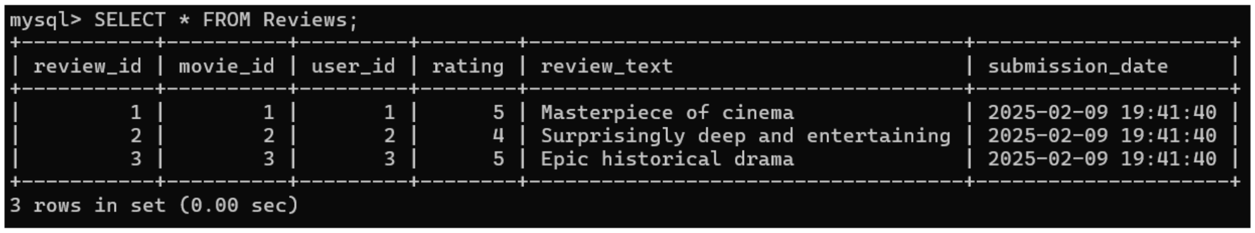
```
mysql> SELECT * FROM Genres;
+----------+------------+
| genre_id | genre_name |
+----------+------------+
|        1 | Action     |
|        2 | Comedy     |
|        3 | Drama      |
+----------+------------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Producers;*



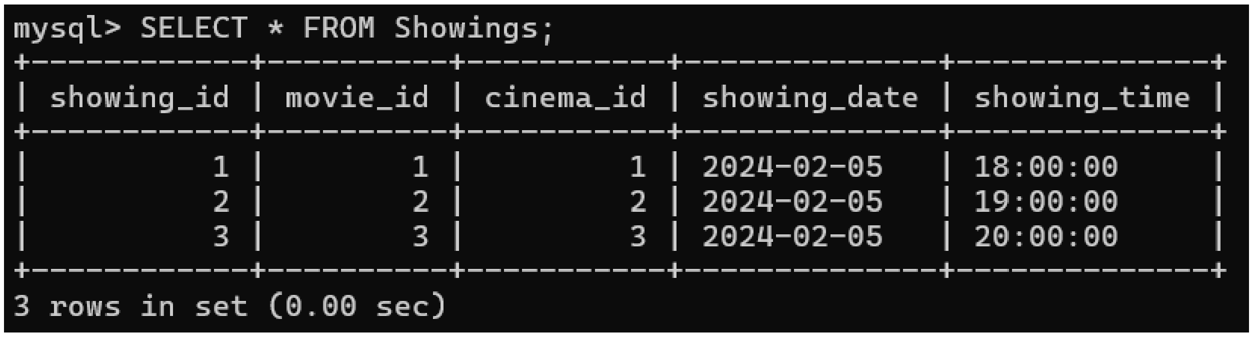
```
mysql> SELECT * FROM Producers;
+-------------+--------------+------------------------------+
| producer_id | name         | contact_info                 |
+-------------+--------------+------------------------------+
|           1 | Emma Thomas  | emma.thomas@syncopy.com      |
|           2 | Tom Ackerley | tom.ackerley@luckychap.com   |
|           3 | Dan Friedkin | dan.friedkin@imperative.com  |
+-------------+--------------+------------------------------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Reviews;*



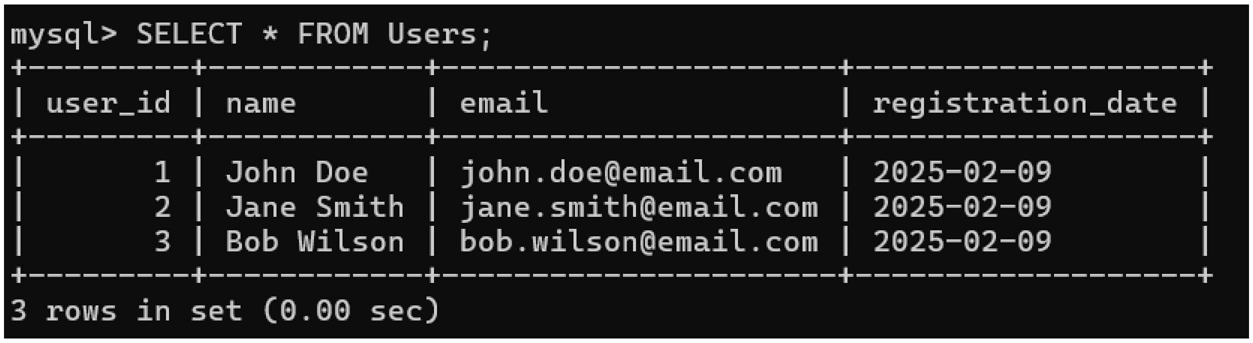
```
mysql> SELECT * FROM Reviews;
+-----------+----------+---------+--------+-----------------------------------+---------------------+
| review_id | movie_id | user_id | rating | review_text                       | submission_date     |
+-----------+----------+---------+--------+-----------------------------------+---------------------+
|         1 |        1 |       1 |      5 | Masterpiece of cinema             | 2025-02-09 19:41:40 |
|         2 |        2 |       2 |      4 | Surprisingly deep and entertaining | 2025-02-09 19:41:40 |
|         3 |        3 |       3 |      5 | Epic historical drama             | 2025-02-09 19:41:40 |
+-----------+----------+---------+--------+-----------------------------------+---------------------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Showings;*



```
mysql> SELECT * FROM Showings;
+------------+----------+-----------+--------------+--------------+
| showing_id | movie_id | cinema_id | showing_date | showing_time |
+------------+----------+-----------+--------------+--------------+
|          1 |        1 |         1 | 2024-02-05   | 18:00:00     |
|          2 |        2 |         2 | 2024-02-05   | 19:00:00     |
|          3 |        3 |         3 | 2024-02-05   | 20:00:00     |
+------------+----------+-----------+--------------+--------------+
3 rows in set (0.00 sec)
```

*SELECT * FROM Users;*



```
mysql> SELECT * FROM Users;
+---------+------------+----------------------+-------------------+
| user_id | name       | email                | registration_date |
+---------+------------+----------------------+-------------------+
|       1 | John Doe   | john.doe@email.com   | 2025-02-09        |
|       2 | Jane Smith | jane.smith@email.com | 2025-02-09        |
|       3 | Bob Wilson | bob.wilson@email.com | 2025-02-09        |
+---------+------------+----------------------+-------------------+
3 rows in set (0.00 sec)
```

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

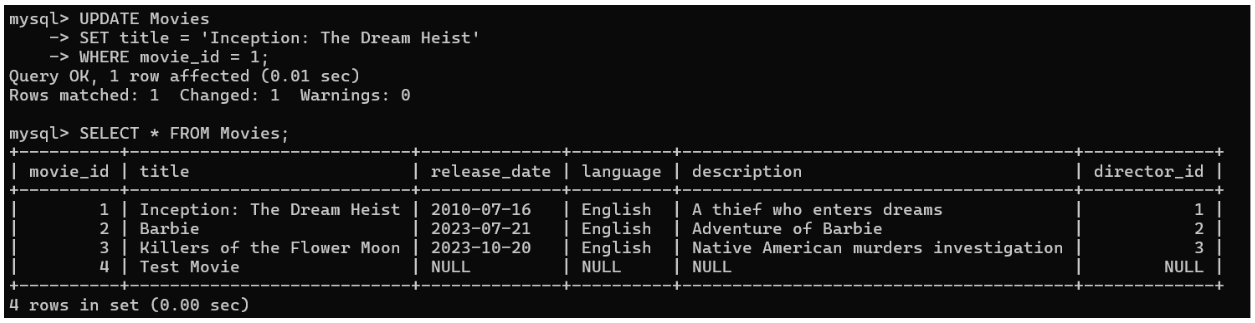
2. Then try each error case separately to observe the error messages:
-- Run each invalid insertion case one by one

3. To verify the data:
-- View inserted data
*SELECT * FROM Movies;*
*SELECT * FROM Actors;*
*SELECT * FROM Directors;*
-- etc.

CODE FOR TASK 3:

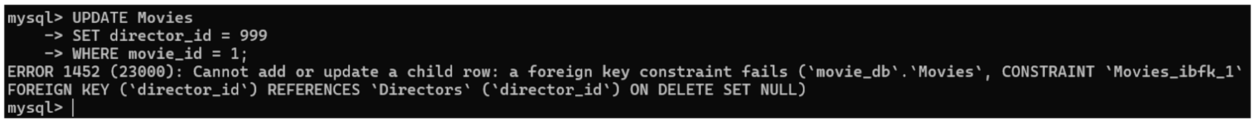
1. UPDATE OPERATIONS:

-- Update 1: Simple update of a movie title
UPDATE Movies
SET title = 'Inception: The Dream Heist'
WHERE movie_id = 1;



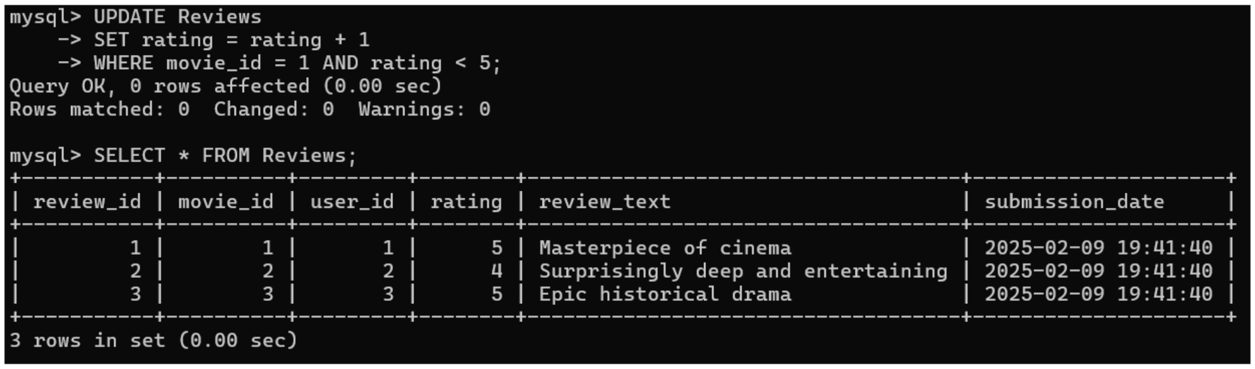
```
mysql> UPDATE Movies
    -> SET title = 'Inception: The Dream Heist'
    -> WHERE movie_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Movies;
+----------+----------------------------+--------------+----------+-----------------------------------------+-------------+
| movie_id | title                      | release_date | language | description                             | director_id |
+----------+----------------------------+--------------+----------+-----------------------------------------+-------------+
|        1 | Inception: The Dream Heist | 2010-07-16   | English  | A thief who enters dreams               |           1 |
|        2 | Barbie                     | 2023-07-21   | English  | Adventure of Barbie                     |           2 |
|        3 | Killers of the Flower Moon | 2023-10-20   | English  | Native American murders investigation   |           3 |
|        4 | Test Movie                 | NULL         | NULL     | NULL                                    |        NULL |
+----------+----------------------------+--------------+----------+-----------------------------------------+-------------+
4 rows in set (0.00 sec)
```

-- 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;



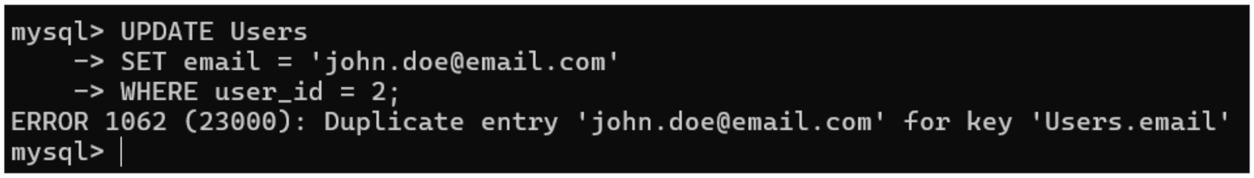
```
mysql> UPDATE Movies
    -> SET director_id = 999
    -> WHERE movie_id = 1;
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`movie_db`.`Movies`, CONSTRAINT `Movies_ibfk_1`
FOREIGN KEY (`director_id`) REFERENCES `Directors` (`director_id`) ON DELETE SET NULL)
mysql> |
```

-- Update 3: Update multiple reviews for a movie
UPDATE Reviews
SET rating = rating + 1
WHERE movie_id = 1 AND rating < 5;



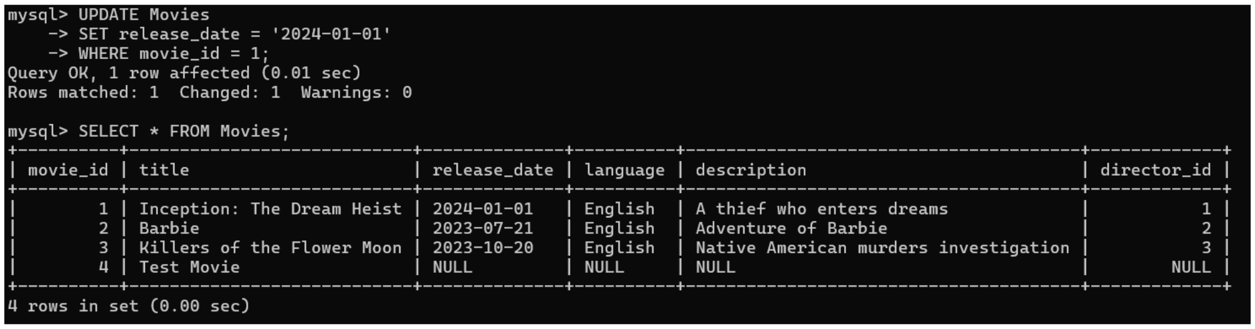
```
mysql> UPDATE Reviews
    -> SET rating = rating + 1
    -> WHERE movie_id = 1 AND rating < 5;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> SELECT * FROM Reviews;
+-----------+----------+---------+--------+-------------------------------------+---------------------+
| review_id | movie_id | user_id | rating | review_text                         | submission_date     |
+-----------+----------+---------+--------+-------------------------------------+---------------------+
|         1 |        1 |       1 |      5 | Masterpiece of cinema               | 2025-02-09 19:41:40 |
|         2 |        2 |       2 |      4 | Surprisingly deep and entertaining  | 2025-02-09 19:41:40 |
|         3 |        3 |       3 |      5 | Epic historical drama               | 2025-02-09 19:41:40 |
+-----------+----------+---------+--------+-------------------------------------+---------------------+
3 rows in set (0.00 sec)
```

-- 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;



```
mysql> UPDATE Users
    -> SET email = 'john.doe@email.com'
    -> WHERE user_id = 2;
ERROR 1062 (23000): Duplicate entry 'john.doe@email.com' for key 'Users.email'
mysql> |
```

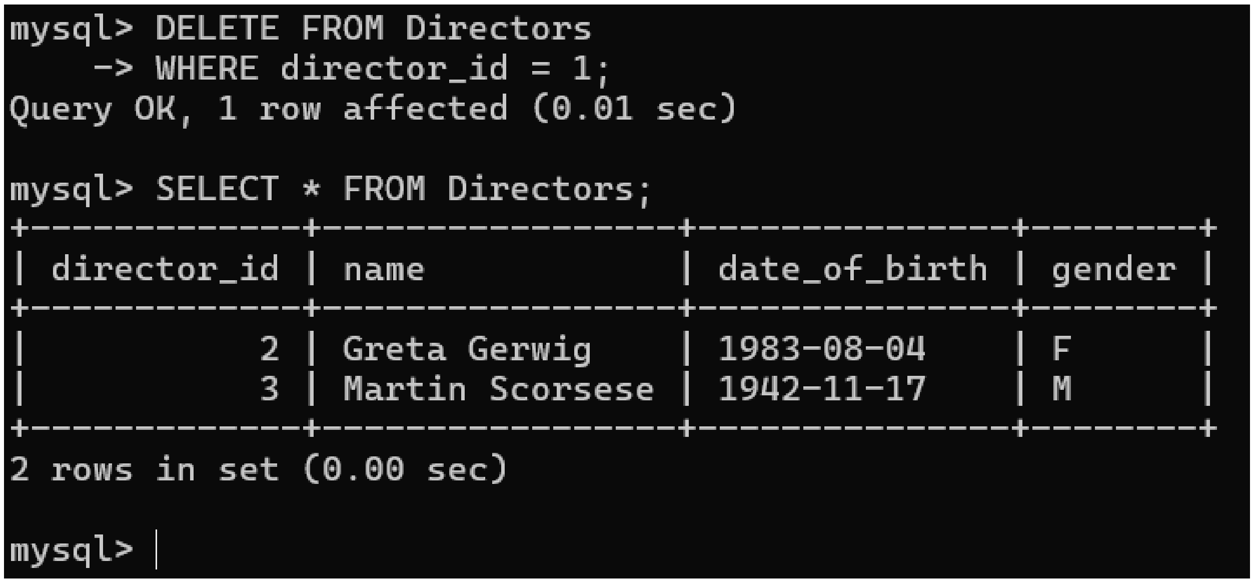
-- Update 5: Update movie release date
UPDATE Movies
SET release_date = '2024-01-01'
WHERE movie_id = 1;



```
mysql> UPDATE Movies
    -> SET release_date = '2024-01-01'
    -> WHERE movie_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Movies;
+----------+-----------------------------+--------------+----------+---------------------------------------+-------------+
| movie_id | title                       | release_date | language | description                           | director_id |
+----------+-----------------------------+--------------+----------+---------------------------------------+-------------+
|        1 | Inception: The Dream Heist  | 2024-01-01   | English  | A thief who enters dreams             |           1 |
|        2 | Barbie                      | 2023-07-21   | English  | Adventure of Barbie                   |           2 |
|        3 | Killers of the Flower Moon  | 2023-10-20   | English  | Native American murders investigation |           3 |
|        4 | Test Movie                  | NULL         | NULL     | NULL                                  |        NULL |
+----------+-----------------------------+--------------+----------+---------------------------------------+-------------+
4 rows in set (0.00 sec)
```

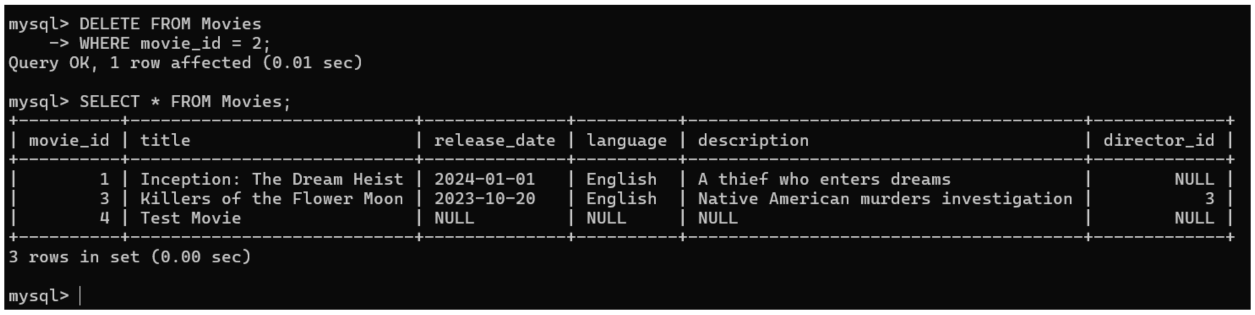
2. DELETE OPERATIONS:

-- 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;



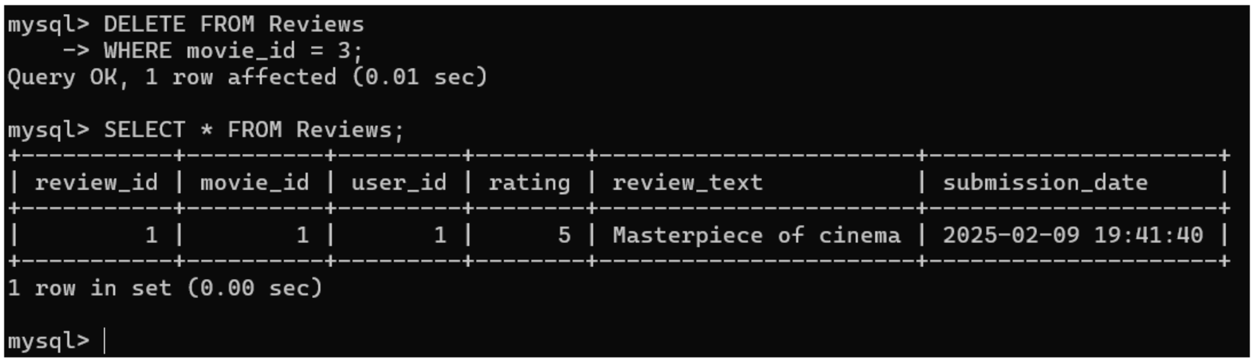
```
mysql> DELETE FROM Directors
    -> WHERE director_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Directors;
+-------------+-----------------+---------------+--------+
| director_id | name            | date_of_birth | gender |
+-------------+-----------------+---------------+--------+
|           2 | Greta Gerwig    | 1983-08-04    | F      |
|           3 | Martin Scorsese | 1942-11-17    | M      |
+-------------+-----------------+---------------+--------+
2 rows in set (0.00 sec)

mysql> 
```

-- Delete 2: Delete a movie (this will cascade)
DELETE FROM Movies
WHERE movie_id = 2;



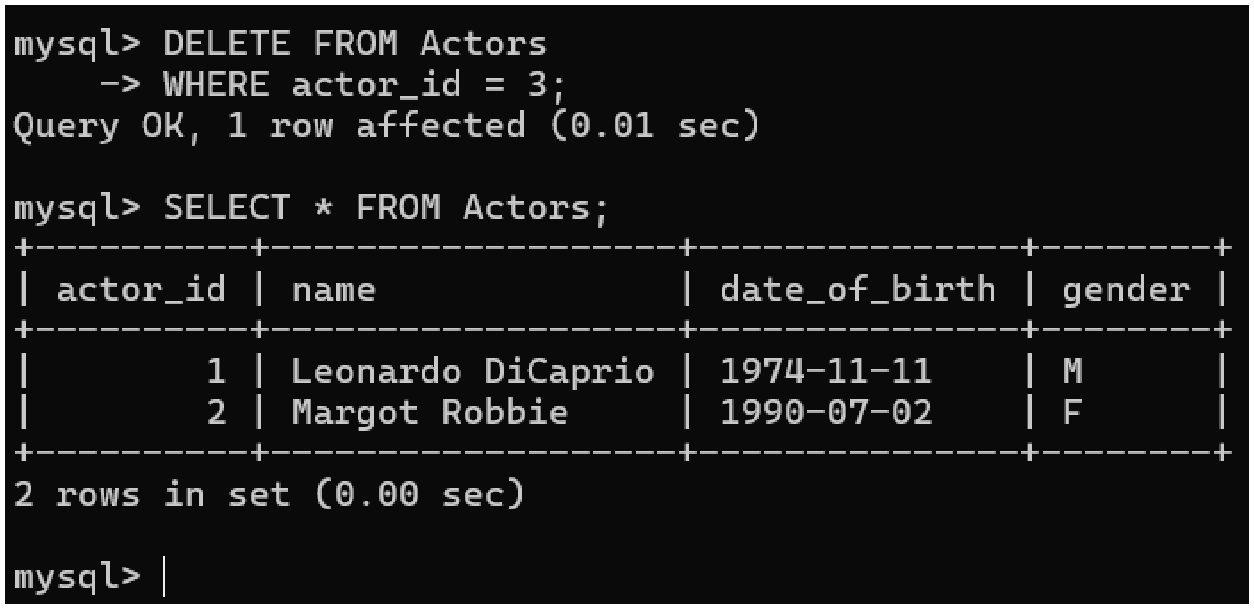
```
mysql> DELETE FROM Movies
    -> WHERE movie_id = 2;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Movies;
+----------+----------------------------+--------------+----------+----------------------------------------+-------------+
| movie_id | title                      | release_date | language | description                            | director_id |
+----------+----------------------------+--------------+----------+----------------------------------------+-------------+
|        1 | Inception: The Dream Heist | 2024-01-01   | English  | A thief who enters dreams              |        NULL |
|        3 | Killers of the Flower Moon | 2023-10-20   | English  | Native American murders investigation |           3 |
|        4 | Test Movie                 | NULL         | NULL     | NULL                                   |        NULL |
+----------+----------------------------+--------------+----------+----------------------------------------+-------------+
3 rows in set (0.00 sec)

mysql> 
```

-- Delete 3: Delete all reviews for a specific movie
DELETE FROM Reviews
WHERE movie_id = 3;



```
mysql> DELETE FROM Reviews
    -> WHERE movie_id = 3;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Reviews;
+-----------+----------+---------+--------+----------------------+---------------------+
| review_id | movie_id | user_id | rating | review_text          | submission_date     |
+-----------+----------+---------+--------+----------------------+---------------------+
|         1 |        1 |       1 |      5 | Masterpiece of cinema | 2025-02-09 19:41:40 |
+-----------+----------+---------+--------+----------------------+---------------------+
1 row in set (0.00 sec)

mysql> |
```

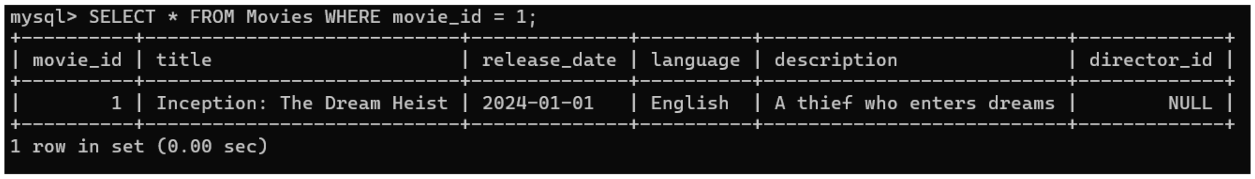
-- Delete 4: Delete an actor (this will affect Movie_Actor junction table due to CASCADE)
DELETE FROM Actors
WHERE actor_id = 3;



```
mysql> DELETE FROM Actors
    -> WHERE actor_id = 3;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Actors;
+----------+-------------------+---------------+--------+
| actor_id | name              | date_of_birth | gender |
+----------+-------------------+---------------+--------+
|        1 | Leonardo DiCaprio | 1974-11-11    | M      |
|        2 | Margot Robbie     | 1990-07-02    | F      |
+----------+-------------------+---------------+--------+
2 rows in set (0.00 sec)

mysql> |
```

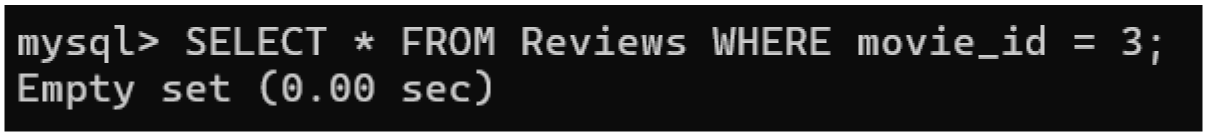
3. VERIFICATION QUERIES:

-- Check updated movie title
*SELECT * FROM Movies WHERE movie_id = 1;*



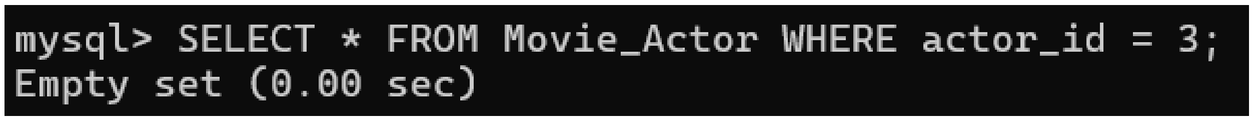
```
mysql> SELECT * FROM Movies WHERE movie_id = 1;
+----------+-----------------------------+--------------+----------+---------------------------+-------------+
| movie_id | title                       | release_date | language | description               | director_id |
+----------+-----------------------------+--------------+----------+---------------------------+-------------+
|        1 | Inception: The Dream Heist  | 2024-01-01   | English  | A thief who enters dreams |        NULL |
+----------+-----------------------------+--------------+----------+---------------------------+-------------+
1 row in set (0.00 sec)
```

-- Check if reviews were deleted (after Delete 3)
*SELECT * FROM Reviews WHERE movie_id = 3;*



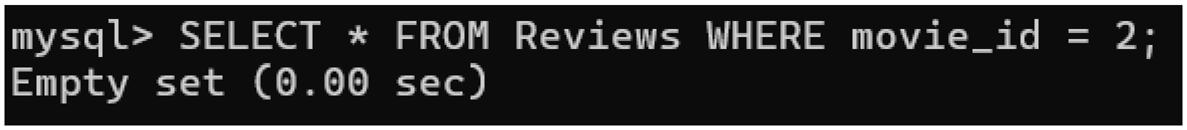
```
mysql> SELECT * FROM Reviews WHERE movie_id = 3;
Empty set (0.00 sec)
```

-- Check if Movie_Actor entries were deleted (after Delete 4)
*SELECT * FROM Movie_Actor WHERE actor_id = 3;*



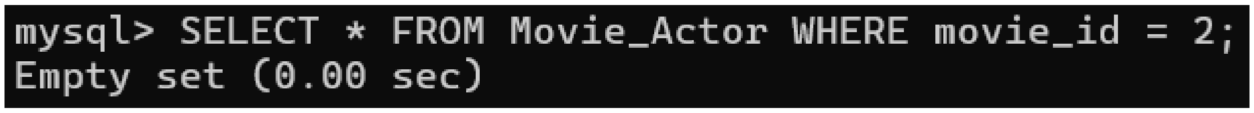
```
mysql> SELECT * FROM Movie_Actor WHERE actor_id = 3;
Empty set (0.00 sec)
```

-- Check if movie deletion cascaded to related tables
*SELECT * FROM Reviews WHERE movie_id = 2;*



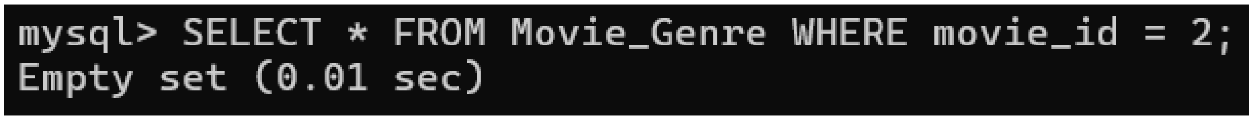
```
mysql> SELECT * FROM Reviews WHERE movie_id = 2;
Empty set (0.00 sec)
```

*SELECT * FROM Movie_Actor WHERE movie_id = 2;*



```
mysql> SELECT * FROM Movie_Actor WHERE movie_id = 2;
Empty set (0.00 sec)
```

*SELECT * FROM Movie_Genre WHERE movie_id = 2;*



```
mysql> SELECT * FROM Movie_Genre WHERE movie_id = 2;
Empty set (0.01 sec)
```

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