CINEMA TICKETS BOOKING SYSTEM USING JAVA

Databaseconnection.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/CinemaBookingSystem";
    private static final String USER = "root";
    private static final String PASSWORD = "Ishu@123";
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

Movie.java

```
public class Movie {
    private int id;
    private String title;
    private String genre;
    private int duration;
    private float rating;
    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getGenre() { return genre; }
    public void setGenre(String genre) { this.genre = genre; }
    public int getDuration() { return duration; }
    public void setDuration(int duration) { this.duration = duration; }
    public float getRating() { return rating; }
    public void setRating(float rating) { this.rating = rating; }
}
```

Customer.java

```
public class Customer {
    private int id;
    private String name;
    private String email;
    private String phone;
    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getPhone() { return phone; }
    public void setPhone(String phone) { this.phone = phone; }
}
```

Show.java

```
public class Show {
    private int id;
    private int movieId;
    private int theaterId;
    private String date;
    private String time;
    private int seatsAvailable;
    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public int getMovieId() { return movieId; }
    public void setMovieId(int movieId) { this.movieId = movieId; }
    public int getTheaterId() { return theaterId; }
    public void setTheaterId(int theaterId) { this.theaterId = theaterId; }
    public String getDate() { return date; }
    public void setDate(String date) { this.date = date; }
    public String getTime() { return time; }
    public void setTime(String time) { this.time = time; }
    public int getSeatsAvailable() { return seatsAvailable; }
    public void setSeatsAvailable(int seatsAvailable) { this.seatsAvailable = seatsAvailable; }
}
```

MovieDAO.java

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class MovieDAO {
    public List<Movie> getAllMovies() throws SQLException {
        List<Movie> movies = new ArrayList<>();
        String query = "SELECT * FROM Movies";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                Movie movie = new Movie();
                movie.setId(rs.getInt("MovieID"));
                movie.setTitle(rs.getString("Title"));
                movie.setGenre(rs.getString("Genre"));
                movie.setDuration(rs.getInt("Duration"));
                movie.setRating(rs.getFloat("Rating"));
                movies.add(movie);
            }
        }
        return movies;
    }
}
```

BookingDAO.java

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class BookingDAO {
    public boolean bookTickets(int customerId, int showId, int seatsToBook) throws SQLException {
        String checkSeatsQuery = "SELECT SeatsAvailable FROM Shows WHERE ShowID = ?";
        String updateSeatsQuery = "UPDATE Shows SET SeatsAvailable = SeatsAvailable - ? WHERE ShowID = ? AND SeatsAvailable >= ?";
        String insertBookingQuery = "INSERT INTO Bookings (CustomerID, ShowID, SeatsBooked, BookingDate) VALUES (?, ?, ?, CURDATE())"
        try (Connection conn = DatabaseConnection.getConnection()) {
            conn.setAutoCommit(false);
            // Check if seats are available
            try (PreparedStatement checkStmt = conn.prepareStatement(checkSeatsQuery)) {
```

```
            checkStmt.setInt(1, showId);
            ResultSet rs = checkStmt.executeQuery();
            if (rs.next()) {
                int seatsAvailable = rs.getInt("SeatsAvailable");
                if (seatsAvailable < seatsToBook) {
                    System.out.println("Not enough seats available.");
                    return false;
                }
            }
        }
        // Update available seats
        try (PreparedStatement updateStmt = conn.prepareStatement(updateSeatsQuery)) {
            updateStmt.setInt(1, seatsToBook);
            updateStmt.setInt(2, showId);
            updateStmt.setInt(3, seatsToBook);
            updateStmt.executeUpdate();
        }
        // Create booking record
        try (PreparedStatement insertStmt = conn.prepareStatement(insertBookingQuery)) {
            insertStmt.setInt(1, customerId);
            insertStmt.setInt(2, showId);
            insertStmt.setInt(3, seatsToBook);
            insertStmt.executeUpdate();
        }
        conn.commit();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
        }
    }
    public boolean cancelBooking(int bookingId) throws SQLException {
        String getSeatsQuery = "SELECT SeatsBooked, ShowID FROM Bookings WHERE BookingID
= ?";
        String updateSeatsQuery = "UPDATE Shows SET SeatsAvailable = SeatsAvailable + ?
WHERE ShowID = ?";
        String deleteBookingQuery = "DELETE FROM Bookings WHERE BookingID = ?";

        try (Connection conn = DatabaseConnection.getConnection()) {
            conn.setAutoCommit(false);
            int seatsToReturn = 0;
            int showId = 0;
            // Get seats booked and show ID
            try (PreparedStatement getSeatsStmt = conn.prepareStatement(getSeatsQuery)) {
                getSeatsStmt.setInt(1, bookingId);
                ResultSet rs = getSeatsStmt.executeQuery();
                if (rs.next()) {
                    seatsToReturn = rs.getInt("SeatsBooked");
                    showId = rs.getInt("ShowID");
                }
            }
            // Update seats available in show
            try (PreparedStatement updateSeatsStmt =
conn.prepareStatement(updateSeatsQuery)) {
                updateSeatsStmt.setInt(1, seatsToReturn);
                updateSeatsStmt.setInt(2, showId);
                updateSeatsStmt.executeUpdate();
            }
            // Delete booking record
            try (PreparedStatement deleteBookingStmt =
conn.prepareStatement(deleteBookingQuery)) {
                deleteBookingStmt.setInt(1, bookingId);
                deleteBookingStmt.executeUpdate();
            }
            conn.commit();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

CinemaApp.java

```
import java.sql.SQLException;
public class CinemaApp {
    public static void main(String[] args) {
        MovieDAO movieDAO = new MovieDAO();
        BookingDAO bookingDAO = new BookingDAO();
        try {
            // View all movies
            System.out.println("Movies:");
            for (Movie movie : movieDAO.getAllMovies()) {
                System.out.println(movie.getTitle());
            }
            // Book tickets
            System.out.println("\nBooking tickets...");
            boolean bookingStatus = bookingDAO.bookTickets(1, 1, 2);
            System.out.println("Booking successful: " + bookingStatus);
            // Cancel booking
            System.out.println("\nCanceling booking...");
            boolean cancelStatus = bookingDAO.cancelBooking(1);
            System.out.println("Cancellation successful: " + cancelStatus);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

SQL code

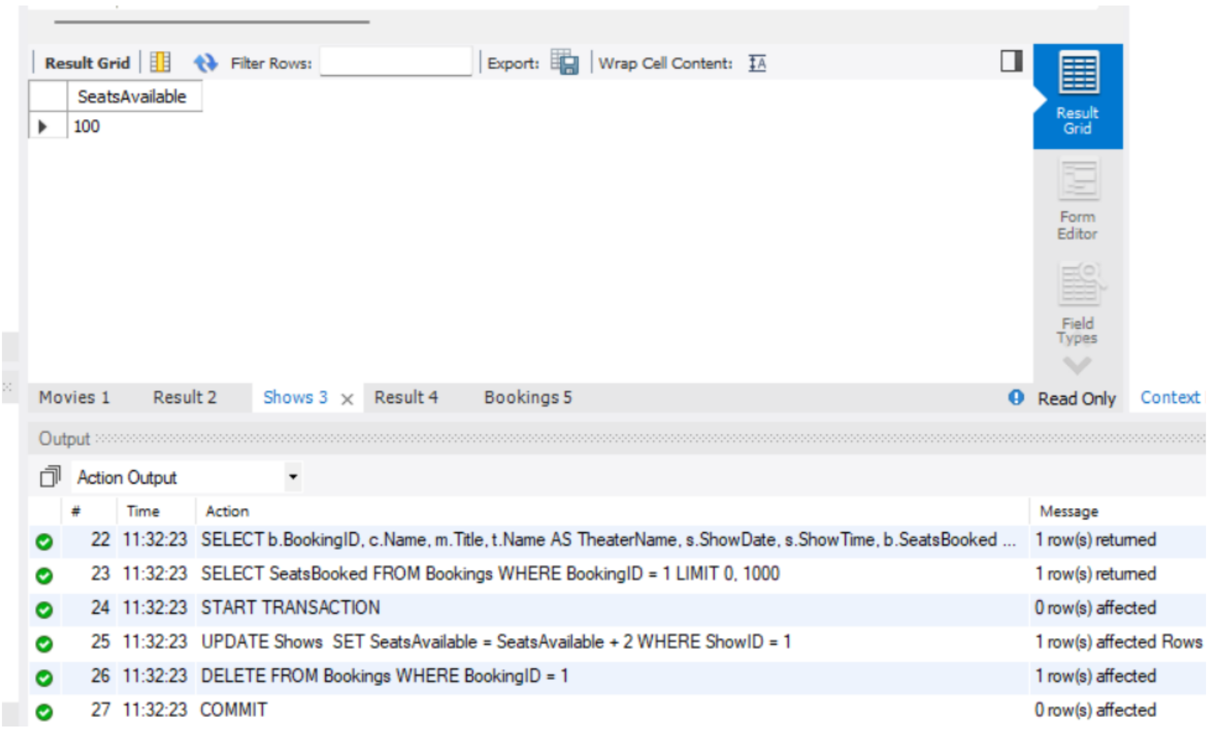
```
-- Create a database for the cinema booking system
USE CinemaBookingSystem;
DROP DATABASE CinemaBookingSystem;
CREATE DATABASE CinemaBookingSystem;
USE CinemaBookingSystem;
-- Create table for movies
CREATE TABLE Movies (
    MovieID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(100) NOT NULL,
    Genre VARCHAR(50),
    Duration INT, -- Duration in minutes
    Rating FLOAT
);
-- Create table for theaters
CREATE TABLE Theaters (
    TheaterID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Location VARCHAR(100)
);
```

```sql
-- Create table for shows
CREATE TABLE Shows (
    ShowID INT PRIMARY KEY AUTO_INCREMENT,
    MovieID INT,
    TheaterID INT,
    ShowDate DATE,
    ShowTime TIME,
    SeatsAvailable INT,
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),
    FOREIGN KEY (TheaterID) REFERENCES Theaters(TheaterID)
);
-- Create table for customers
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(15)
);
-- Create table for bookings
CREATE TABLE Bookings (
    BookingID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT,
    ShowID INT,
    SeatsBooked INT,
    BookingDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (ShowID) REFERENCES Shows(ShowID)
);
-- Insert sample movies
```

```sql
INSERT INTO Movies (Title, Genre, Duration, Rating)

VALUES

    ('Avengers: Endgame', 'Action', 181, 8.4),

    ('Inception', 'Sci-Fi', 148, 8.8),

    ('The Godfather', 'Crime', 175, 9.2);

-- Insert sample theaters

INSERT INTO Theaters (Name, Location)

VALUES

    ('Cinema City', 'Downtown'),

    ('Grand Theater', 'Uptown');

-- Insert sample shows

INSERT INTO Shows (MovieID, TheaterID, ShowDate, ShowTime, SeatsAvailable)

VALUES

    (1, 1, '2024-11-15', '18:00:00', 100),

    (2, 1, '2024-11-16', '21:00:00', 80),

    (3, 2, '2024-11-17', '19:30:00', 50);

-- Insert sample customers

INSERT INTO Customers (Name, Email, Phone)

VALUES

    ('John Doe', 'john@example.com', '123-456-7890'),

    ('Jane Smith', 'jane@example.com', '098-765-4321');

SELECT * FROM Movies;

SELECT s.ShowID, m.Title, t.Name AS TheaterName, s.ShowDate, s.ShowTime, s.SeatsAvailable

FROM Shows s

JOIN Movies m ON s.MovieID = m.MovieID

JOIN Theaters t ON s.TheaterID = t.TheaterID

WHERE m.Title = 'Inception';

-- First, check if there are enough seats available

SELECT SeatsAvailable FROM Shows WHERE ShowID = 1;
```

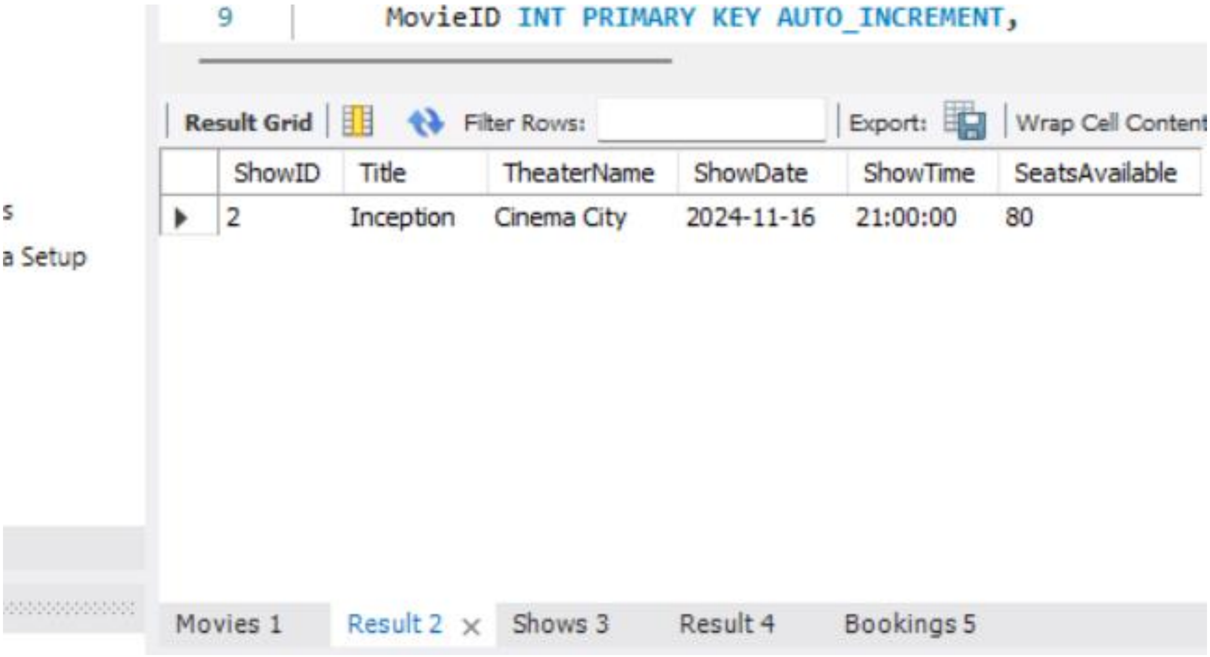
```
-- If enough seats are available, proceed with booking
START TRANSACTION;
-- Update the seats available
UPDATE Shows
SET SeatsAvailable = SeatsAvailable - 2
WHERE ShowID = 1 AND SeatsAvailable >= 2;
-- Insert a new booking
INSERT INTO Bookings (CustomerID, ShowID, SeatsBooked, BookingDate)
VALUES (1, 1, 2, CURDATE());
COMMIT;
SELECT b.BookingID, c.Name, m.Title, t.Name AS TheaterName, s.ShowDate, s.ShowTime, b.SeatsBooked
FROM Bookings b
JOIN Customers c ON b.CustomerID = c.CustomerID
JOIN Shows s ON b.ShowID = s.ShowID
JOIN Movies m ON s.MovieID = m.MovieID
JOIN Theaters t ON s.TheaterID = t.TheaterID
WHERE c.CustomerID = 1;
-- First, retrieve the seats booked for the booking
SELECT SeatsBooked FROM Bookings WHERE BookingID = 1;
-- Begin a transaction to safely cancel the booking
START TRANSACTION;
-- Update the seats available for the show
UPDATE Shows
SET SeatsAvailable = SeatsAvailable + 2
WHERE ShowID = 1;
-- Delete the booking record
DELETE FROM Bookings WHERE BookingID = 1;
COMMIT;
```

OUTPUT



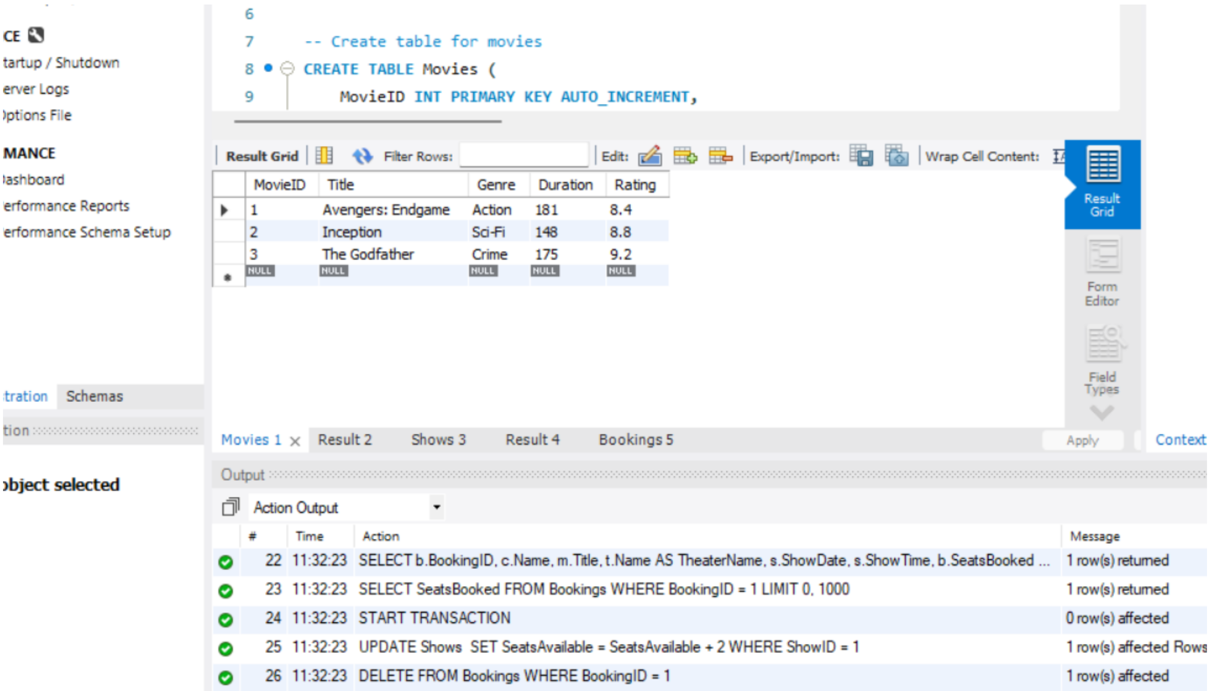
SeatsAvailable
100

	#	Time	Action	Message
✓	22	11:32:23	SELECT b.BookingID, c.Name, m.Title, t.Name AS TheaterName, s.ShowDate, s.ShowTime, b.SeatsBooked ...	1 row(s) returned
✓	23	11:32:23	SELECT SeatsBooked FROM Bookings WHERE BookingID = 1 LIMIT 0, 1000	1 row(s) returned
✓	24	11:32:23	START TRANSACTION	0 row(s) affected
✓	25	11:32:23	UPDATE Shows SET SeatsAvailable = SeatsAvailable + 2 WHERE ShowID = 1	1 row(s) affected Rows
✓	26	11:32:23	DELETE FROM Bookings WHERE BookingID = 1	1 row(s) affected
✓	27	11:32:23	COMMIT	0 row(s) affected



```
    MovieID INT PRIMARY KEY AUTO_INCREMENT,
```

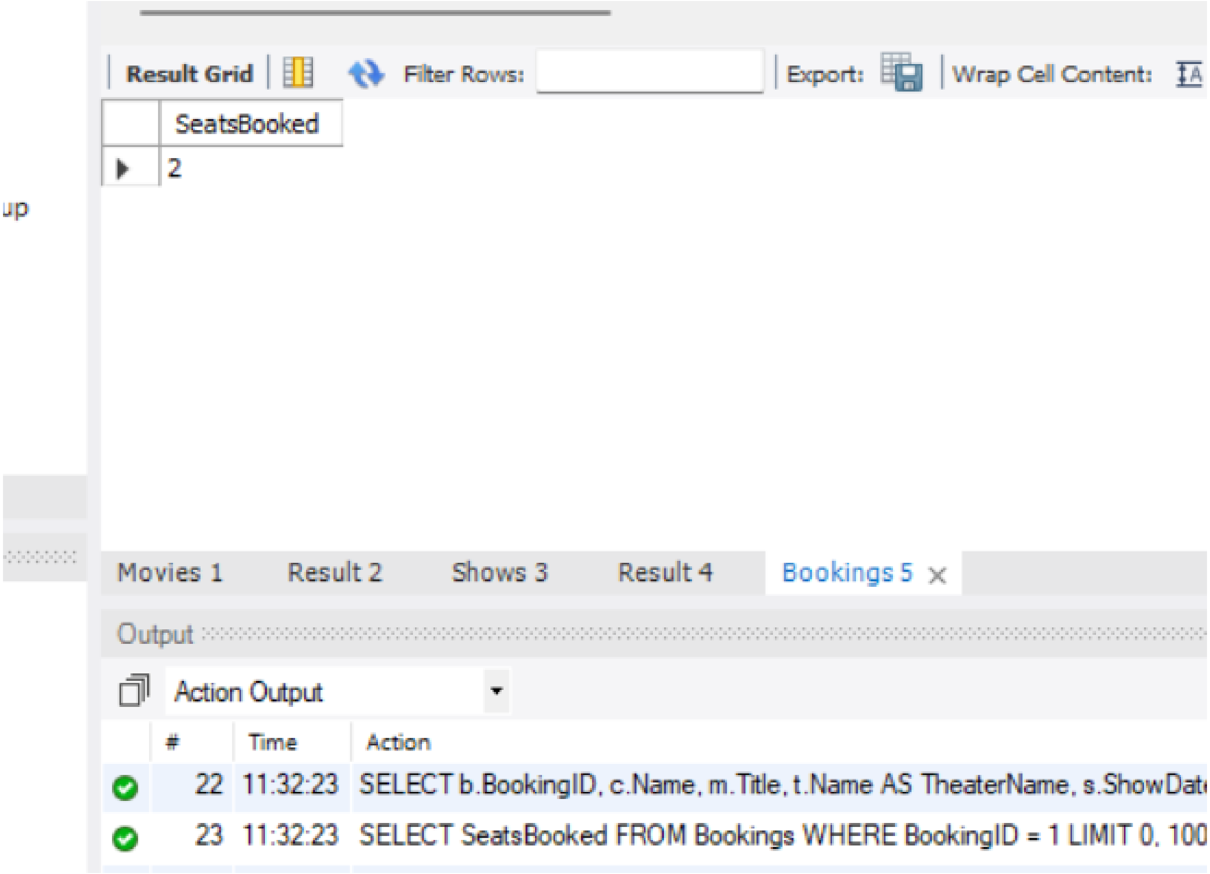
ShowID	Title	TheaterName	ShowDate	ShowTime	SeatsAvailable
2	Inception	Cinema City	2024-11-16	21:00:00	80



```sql
-- Create table for movies
CREATE TABLE Movies (
    MovieID INT PRIMARY KEY AUTO_INCREMENT,
```

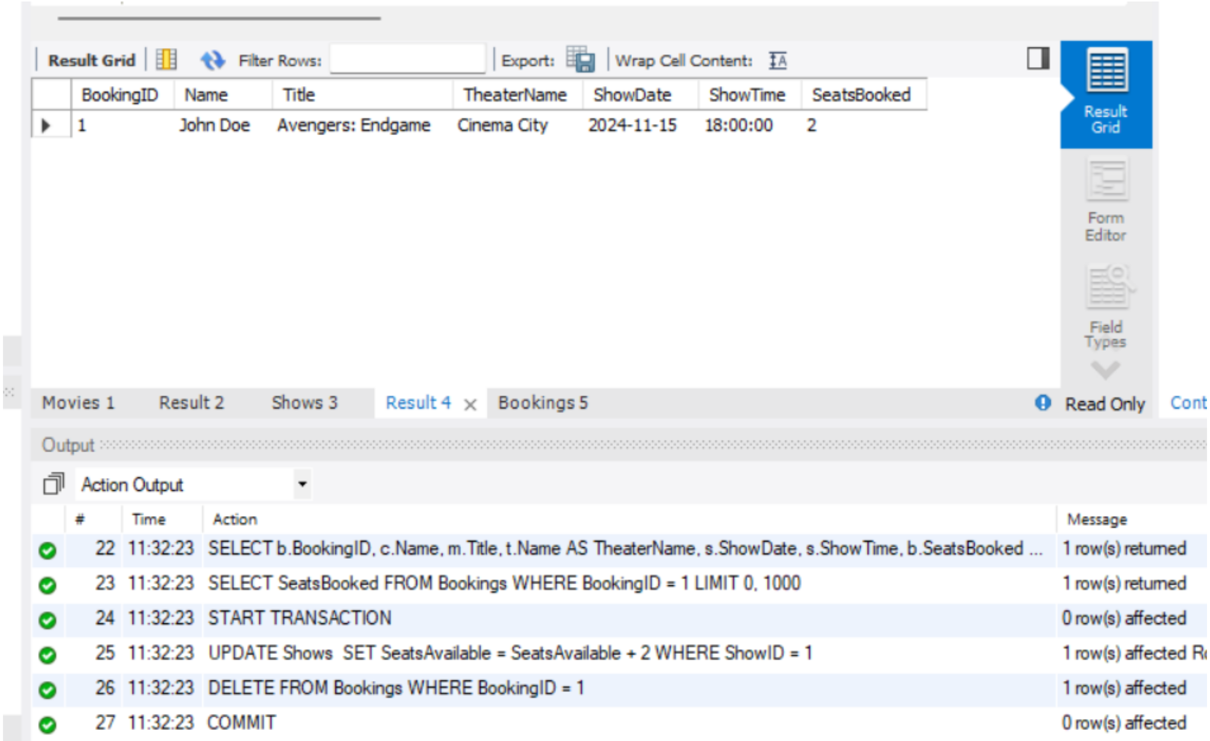
MovieID	Title	Genre	Duration	Rating
1	Avengers: Endgame	Action	181	8.4
2	Inception	Sci-Fi	148	8.8
3	The Godfather	Crime	175	9.2
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                                                   Message
22  11:32:23  SELECT b.BookingID, c.Name, m.Title, t.Name AS TheaterName, s.ShowDate, s.ShowTime, b.SeatsBooked ...   1 row(s) returned
23  11:32:23  SELECT SeatsBooked FROM Bookings WHERE BookingID = 1 LIMIT 0, 1000                                       1 row(s) returned
24  11:32:23  START TRANSACTION                                                                                        0 row(s) affected
25  11:32:23  UPDATE Shows  SET SeatsAvailable = SeatsAvailable + 2 WHERE ShowID = 1                                   1 row(s) affected Rows
26  11:32:23  DELETE FROM Bookings WHERE BookingID = 1                                                                 1 row(s) affected
```



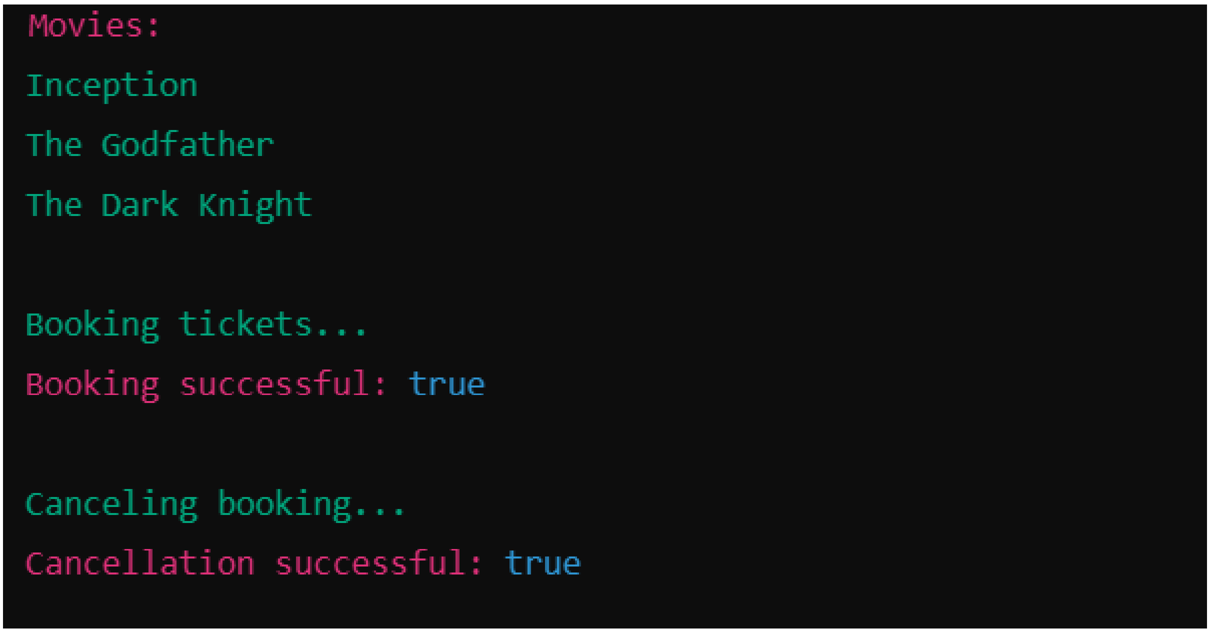
SeatsBooked
2

```
#   Time      Action
22  11:32:23  SELECT b.BookingID, c.Name, m.Title, t.Name AS TheaterName, s.ShowDat
23  11:32:23  SELECT SeatsBooked FROM Bookings WHERE BookingID = 1 LIMIT 0, 100
```



BookingID	Name	Title	TheaterName	ShowDate	ShowTime	SeatsBooked
1	John Doe	Avengers: Endgame	Cinema City	2024-11-15	18:00:00	2

#	Time	Action	Message
22	11:32:23	SELECT b.BookingID, c.Name, m.Title, t.Name AS TheaterName, s.ShowDate, s.ShowTime, b.SeatsBooked ...	1 row(s) returned
23	11:32:23	SELECT SeatsBooked FROM Bookings WHERE BookingID = 1 LIMIT 0, 1000	1 row(s) returned
24	11:32:23	START TRANSACTION	0 row(s) affected
25	11:32:23	UPDATE Shows SET SeatsAvailable = SeatsAvailable + 2 WHERE ShowID = 1	1 row(s) affected R
26	11:32:23	DELETE FROM Bookings WHERE BookingID = 1	1 row(s) affected
27	11:32:23	COMMIT	0 row(s) affected



```
Movies:
Inception
The Godfather
The Dark Knight

Booking tickets...
Booking successful: true

Canceling booking...
Cancellation successful: true
```