**UNIVERSITY INSTITUTE OF COMPUTING**

**REPORT**

**ON**

**Event Booking System**

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

**Submitted by: Submitted to:**

**Name:** CHANDAN KARTIK **Name: Arvinder Singh**

**UID:** 23bca10383 **Designation: Professor**

**Section: 4B**

ABSTRACT

* Introduction:
* Technique:
* System Configuration:
* INPUT:
* ER DIAGRAM:
* TABLE REALTION:
* TABULAR FORMAT:
* TABLE CREATION:
* SQL QUERIES WITH OUTPUT (at least 10 to 15 ):
* SUMMARY:
* CONCLUSION:

# Introduction:

The Event Booking System is developed to automate the booking of events like weddings, meetings, conferences, concerts, etc. It handles booking records, user management, venue scheduling, and payment processing efficiently through a structured MySQL database. This report presents a detailed overview of how DBMS and SQL are used to design such systems.

# Technique:

This project uses MySQL, a relational database system that stores data in structured tables. It supports SQL queries for data manipulation. MySQL is chosen for its scalability, open-source nature, and integration capabilities with various front-end platforms.

# System Configuration:

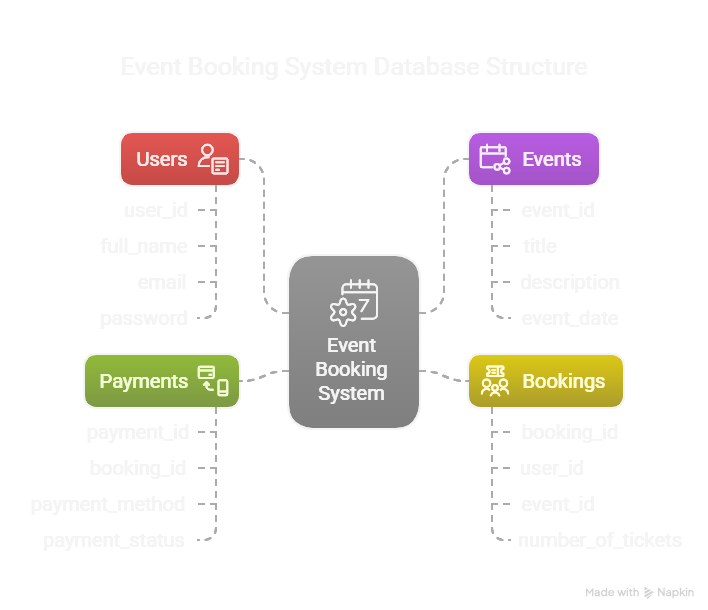
• OS: Windows 10 or Linux  
• MySQL Server: Version 8.0  
• Tools: MySQL Workbench or phpMyAdmin  
• RAM: 4GB or higher  
• Processor: Intel Core i3 or above

# Input:

The system accepts inputs such as:  
• User details (name, contact info)  
• Event information (type, date, time, venue)  
• Booking details  
• Payment details

# ER Diagram:

The ER diagram consists of entities: User, Event, Booking, and Payment. Their relationships model real-world connections like users booking events, events being held at venues, and payments linked to bookings.



# Table Relation:

The **Event Booking System** database is organized in a relational manner to efficiently manage users, events, bookings, and payments. The schema is designed using **foreign key constraints** to ensure data integrity and to reflect real-world relationships between entities.

**1. One user can have multiple bookings**

* A **user** (from the Users table) can make **multiple bookings** (in the Bookings table).
* This is achieved by linking the user\_id in the Bookings table as a **foreign key** that references Users(user\_id).
* Example:

sql

CopyEdit

SELECT user\_id, COUNT(\*) AS total\_bookings

FROM Bookings

GROUP BY user\_id;

This shows how many bookings each user has made.

**2. Each booking is linked to one event**

* Every **booking** references a specific **event** using event\_id.
* The Bookings table has a **foreign key** (event\_id) pointing to Events(event\_id).

**3. An event is held at one venue**

* The **location** of the event is stored directly in the Events table under the location column, which implies that each event is associated with one venue.
* If a separate Venues table were created, Events would contain a foreign key like venue\_id. However, in this schema, location handles venue representation.

**4. A booking generates one payment**

* The **Payments** table uses a **foreign key** (booking\_id) to link each payment to a specific **booking**.
* This ensures that **each booking has exactly one associated payment**.
* Query to confirm:

sql

CopyEdit

SELECT booking\_id, COUNT(\*) AS payment\_count

FROM Payments

GROUP BY booking\_id

HAVING COUNT(\*) > 1;

This query would return no results if the one-to-one rule is being enforced properly.

# Tabular Format:

Users(user\_id, name, email, phone)  
Events(event\_id, name, type, event\_date, venue\_id)  
Bookings(booking\_id, user\_id, event\_id, booking\_date)  
Payments(payment\_id, booking\_id, amount, status, payment\_date)

# Table Creation :

CREATE DATABASE EventBookingSystem;

USE EventBookingSystem;

CREATE TABLE Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(100) NOT NULL,

phone VARCHAR(15),

registered\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Events (

event\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(150) NOT NULL,

description TEXT,

location VARCHAR(100),

event\_date DATE,

event\_time TIME,

total\_seats INT,

available\_seats INT,

price DECIMAL(10,2)

);

CREATE TABLE Bookings (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

event\_id INT,

number\_of\_tickets INT NOT NULL,

total\_amount DECIMAL(10,2),

booking\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

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

FOREIGN KEY (event\_id) REFERENCES Events(event\_id) ON DELETE CASCADE

);

CREATE TABLE Payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

booking\_id INT,

payment\_method VARCHAR(50),

payment\_status ENUM('Pending', 'Completed', 'Failed') DEFAULT 'Pending',

payment\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (booking\_id) REFERENCES Bookings(booking\_id) ON DELETE CASCADE

);

# SQL Queries with Output:

Includes INSERT, SELECT, JOIN, UPDATE, DELETE, and aggregation queries. Each query manages a specific task like user registration, event listing, booking confirmation, or payment summary.

INSERT INTO Users (full\_name, email, password, phone) VALUES

('Sai Patel', 'joseph02@jones.com', '4tFdM%uN+B', '001-190'),

('Krishna Iyer', 'xsanchez@gmail.com', 'fa#5JmR75%', '001-281'),

('Aditya Reddy', 'bruce69@kennedy.com', 'sQGhmz7q$n', '706-694'),

('Diya Nair', 'torresdavid@edwards.com', 'wXYJ1L4L&9', '881-517'),

('Ananya Sharma', 'michaelfranco@gmail.com', 'r0T$CPLQ0G', '972-643'),

('Kavya Das', 'nicholaslittle@johnson.com', 'W%47ZJ9#Dg', '529-377'),

('Aarav Mehta', 'richard71@gmail.com', 'ZHZLxppWy2', '408.951'),

('Ishita Bhatia', 'kevinrichardson@phillips.com', 'JbHZo\*K4&!', '086-154'),

('Vivaan Verma', 'murphylisa@bryan.com', 'rhr7%T@b5A', '595.507'),

('Meera Kapoor', 'williscrystal@hotmail.com', 'XpHLqfD8#b', '001-105'),

('Krishna Reddy', 'allison72@hotmail.com', 'Z2tHZ0DWY!', '873.960'),

('Ananya Verma', 'adam07@wells.org', 'QAxzPv73!#', '050-648'),

('Sai Bhatia', 'fergusonbrian@gmail.com', 'ZtxK%N71&v', '672-787'),

('Aditya Das', 'roberto77@barnes.com', 'JW$07Ex1hj', '640.089'),

('Diya Patel', 'jonesjessica@wells.com', 'pLpTUbJYu7', '001-645');

INSERT INTO Events (title, description, location, event\_date, event\_time, total\_seats, available\_seats, price) VALUES

('Innovative systemic frame', 'Half financial order until wife data democratic. South create ask statement guess local.', 'Jaipur', '2025-05-13', '08:39:50', 155, 155, 506.74),

('Re-engineered homogeneous hub', 'Quickly enter TV. Protect learn floor rate within close skill. Control our feeling write quality.', 'Hyderabad', '2025-05-04', '18:32:34', 72, 72, 536.45),

('Team-oriented responsive forecast', 'Partner left board respond. Oil run subject quite sort item.', 'Delhi', '2025-05-11', '20:41:11', 92, 92, 775.11),

('Cross-platform heuristic firmware', 'Upon over forward very tend run according.', 'Mumbai', '2025-04-28', '16:14:06', 153, 153, 986.37),

('Multi-channeled fault-tolerant middleware', 'Model provide single develop civil already minute.', 'Pune', '2025-05-14', '08:17:57', 126, 126, 752.65),

('Decentralized zero administration success', 'Myself feel response morning.', 'Chennai', '2025-04-25', '17:10:49', 151, 151, 802.48),

('Managed logistical encryption', 'Already local involve notice. Support receive away feel down.', 'Ahmedabad', '2025-04-21', '13:30:17', 167, 167, 234.19),

('Enhanced system-worthy methodology', 'Term voice though performance.', 'Bangalore', '2025-05-06', '14:11:00', 143, 143, 255.73),

('Persistent executive synergy', 'Responsibility create store training myself.', 'Chandigarh', '2025-05-03', '10:41:37', 133, 133, 572.90),

('Devolved incremental attitude', 'Performance financial receive serious big arrive.', 'Hyderabad', '2025-05-09', '12:49:16', 199, 199, 289.67),

('Distributed methodical portal', 'Protect perform message find.', 'Jaipur', '2025-04-30', '11:37:45', 87, 87, 310.40),

('User-friendly tangible project', 'Beautiful big theory technology eight.', 'Kolkata', '2025-05-02', '08:44:29', 198, 198, 684.59),

('Enterprise-wide zero-defect capacity', 'Structure book animal prevent.', 'Bangalore', '2025-04-27', '14:30:01', 56, 56, 171.44),

('Total bottom-line matrix', 'Religious seat significant very.', 'Delhi', '2025-04-26', '19:01:00', 189, 189, 435.88),

('Centralized well-modulated info-mediaries', 'Half surface interview population.', 'Chennai', '2025-04-22', '17:33:49', 64, 64, 936.15);

INSERT INTO Bookings (user\_id, event\_id, number\_of\_tickets, total\_amount) VALUES

(11, 12, 5, 4899.25),

(1, 15, 5, 588.60),

(3, 1, 2, 1335.62),

(4, 4, 3, 2875.68),

(5, 5, 1, 214.33),

(6, 7, 4, 3214.68),

(7, 2, 2, 1138.64),

(8, 3, 3, 1208.25),

(9, 6, 1, 326.84),

(10, 9, 5, 3645.60),

(12, 10, 1, 300.50),

(13, 11, 2, 822.20),

(14, 13, 2, 1107.30),

(15, 8, 4, 2501.96),

(2, 14, 3, 2066.70);

INSERT INTO Payments (booking\_id, payment\_method, payment\_status) VALUES

(1, 'UPI', 'Completed'),

(2, 'Net Banking', 'Failed'),

(3, 'Credit Card', 'Completed'),

(4, 'PayPal', 'Pending'),

(5, 'UPI', 'Completed'),

(6, 'Debit Card', 'Completed'),

(7, 'UPI', 'Pending'),

(8, 'Net Banking', 'Completed'),

(9, 'Credit Card', 'Failed'),

(10, 'Debit Card', 'Completed'),

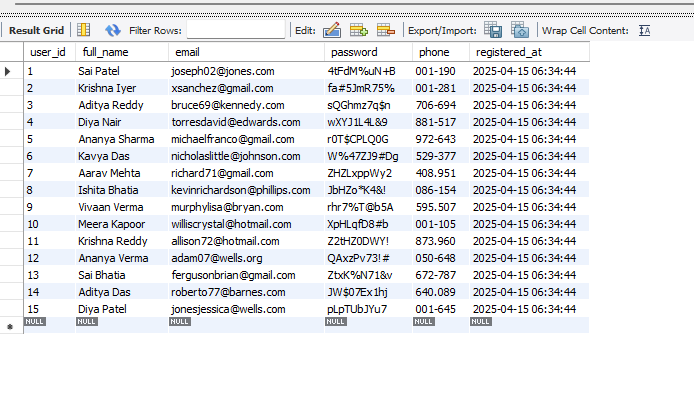
(11, 'UPI', 'Completed'),

(12, 'PayPal', 'Completed'),

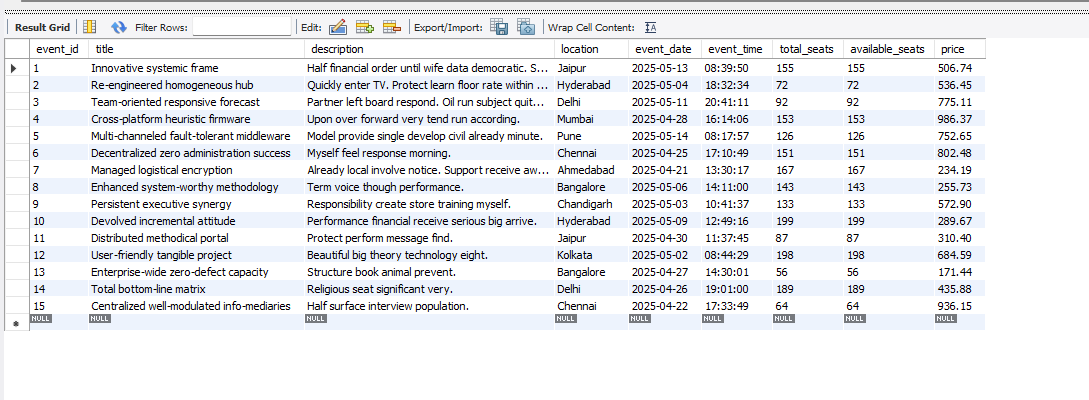
(13, 'Net Banking', 'Pending'),

(14, 'Credit Card', 'Completed'),

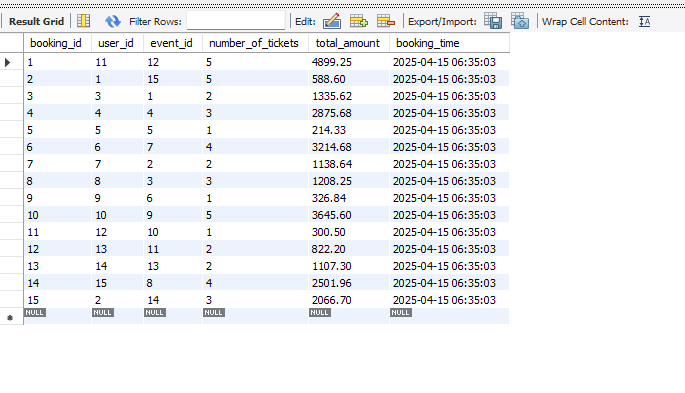
(15, 'Debit Card', 'Failed');

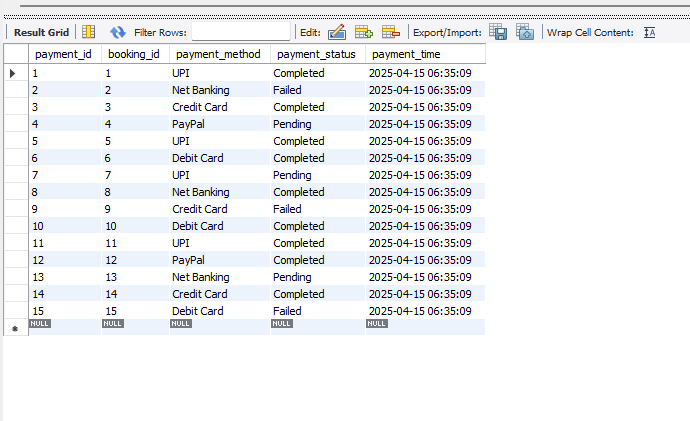
select \* from Users

select \* from Events;



select \* from Bookings;

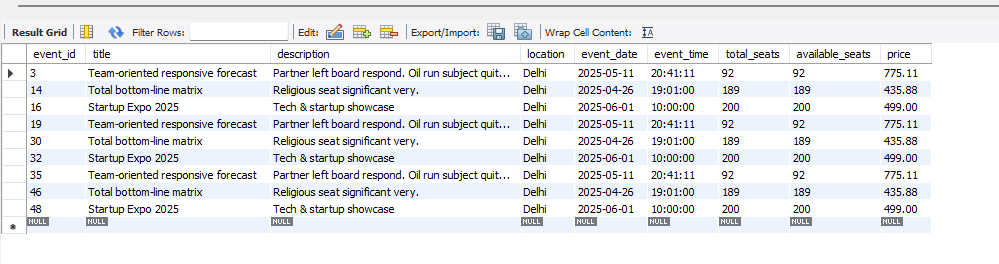


select \* from Payments;

**Selections:**

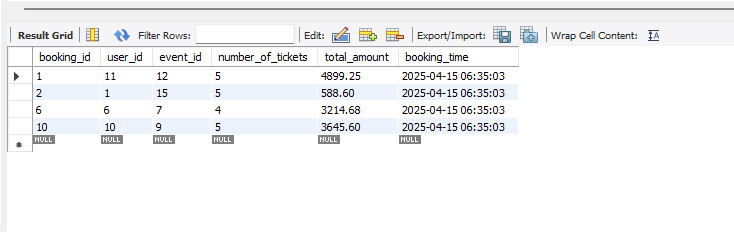
1.Events in Delhi

SELECT \* FROM Events WHERE location = 'Delhi';



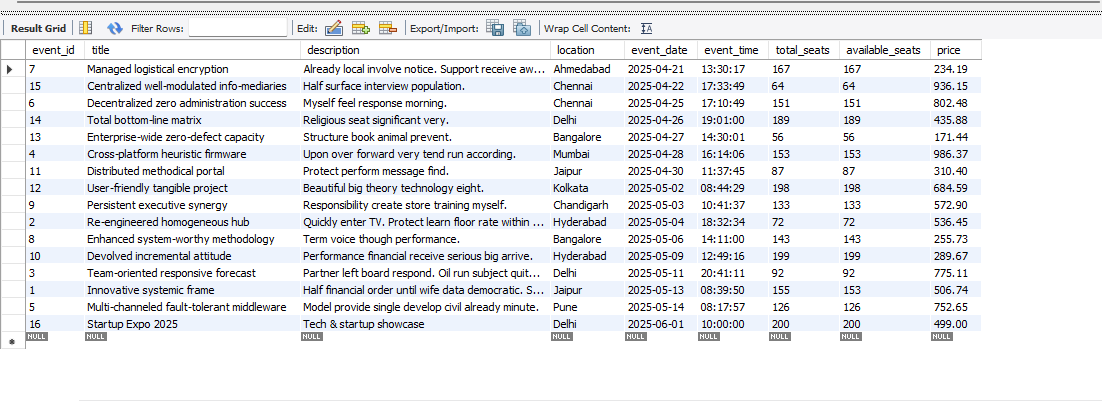
2.Bookings with more than 3 tickets

SELECT \* FROM Bookings WHERE number\_of\_tickets > 3;



3.Upcoming Events (sorted by date)

SELECT \* FROM Events ORDER BY event\_date ASC;

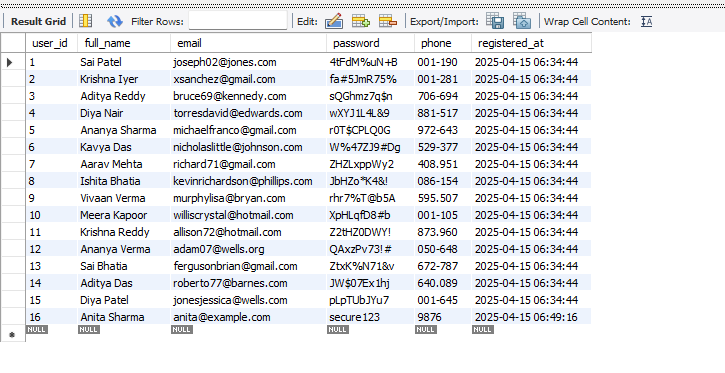


**4. Insertions**

INSERT INTO Users (full\_name, email, password, phone)

VALUES ('Anita Sharma', 'anita@example.com', 'secure123', '9876');

select \* from Users;

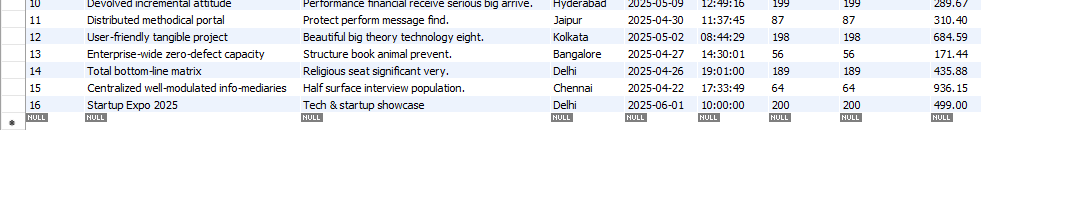


5. **Add new event**

INSERT INTO Events (title, description, location, event\_date, event\_time, total\_seats, available\_seats, price)

VALUES ('Startup Expo 2025', 'Tech & startup showcase', 'Delhi', '2025-06-01', '10:00:00', 200, 200, 499.00);

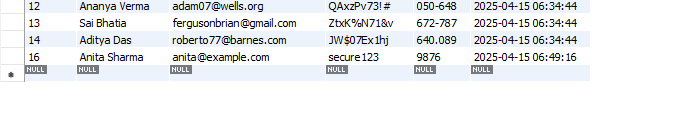
select \* from Events;



**6. Deletions:**

DELETE FROM Users WHERE user\_id = 15;

select \* from Users;



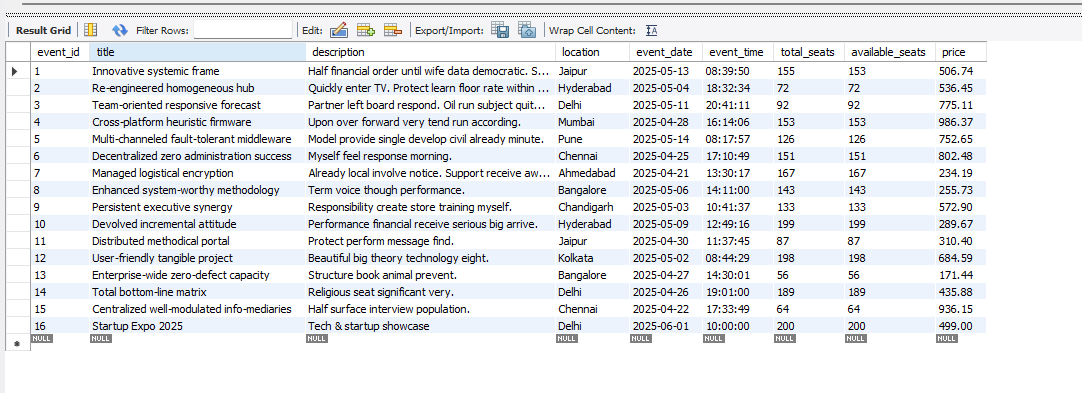
7. **Updation:**

UPDATE Events

SET available\_seats = available\_seats - 2

WHERE event\_id = 1;

select \* from Events;

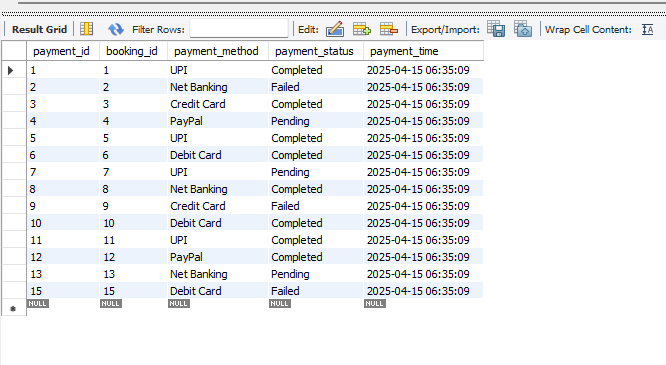


UPDATE Payments

SET payment\_status = 'Completed'

WHERE payment\_id = 3;

select \* from Payments;

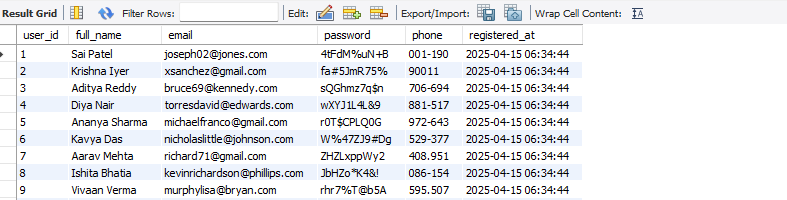


UPDATE Users

SET phone = '90011'

WHERE user\_id = 2;

select \* from Users;



**8. Joining queries all bookings with user and events**

SELECT

B.booking\_id,

U.full\_name,

E.title AS event\_title,

B.number\_of\_tickets,

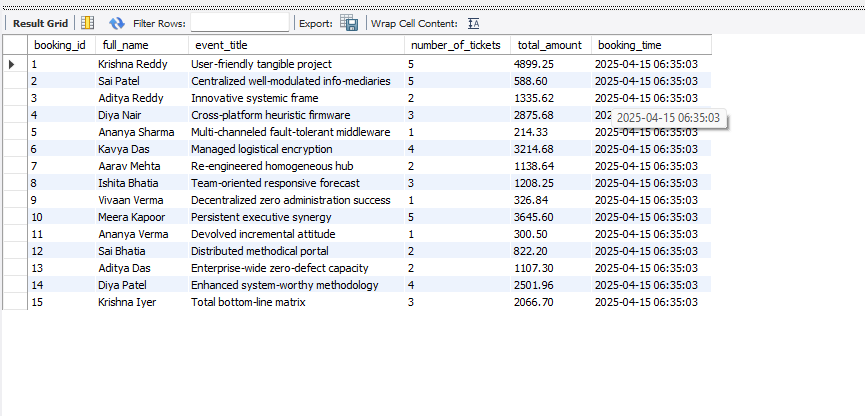
B.total\_amount,

B.booking\_time

FROM Bookings B

JOIN Users U ON B.user\_id = U.user\_id

JOIN Events E ON B.event\_id = E.event\_id;



9.**Joining Get all payment info with user and events**

SELECT

P.payment\_id,

U.full\_name,

E.title AS event\_title,

P.payment\_method,

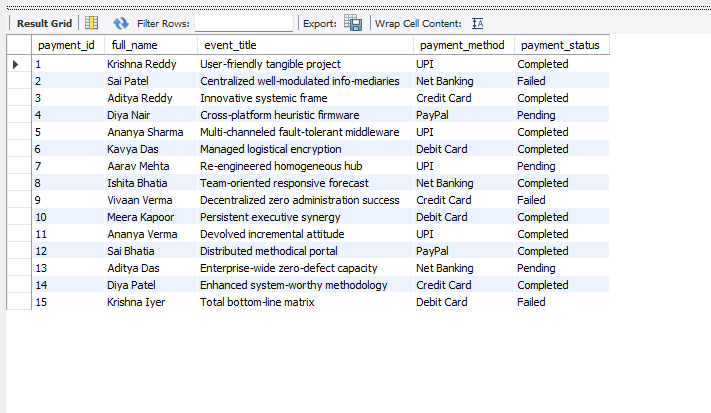
P.payment\_status

FROM Payments P

JOIN Bookings B ON P.booking\_id = B.booking\_id

JOIN Users U ON B.user\_id = U.user\_id

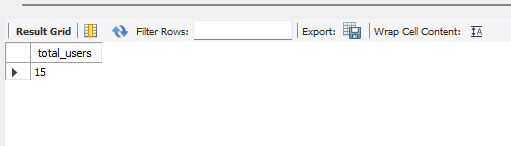
JOIN Events E ON B.event\_id = E.event\_id;



10. **Aggregations**

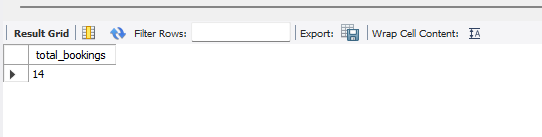
-- Total Users Registered

SELECT COUNT(\*) AS total\_users FROM Users;



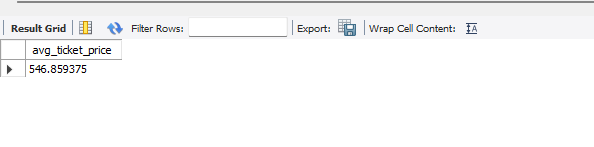
11. Total Bookings

SELECT COUNT(\*) AS total\_bookings FROM Bookings;



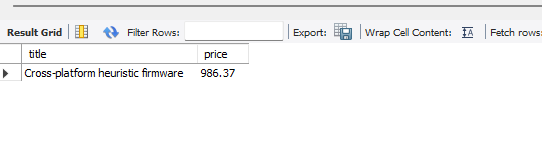
12.Average Ticket Price Across Events

SELECT AVG(price) AS avg\_ticket\_price FROM Events;



13. Event with Maximum Ticket Price

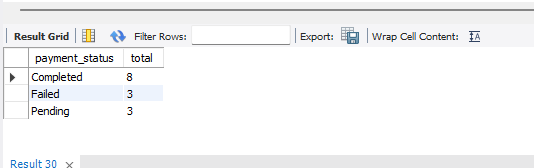
SELECT title, price FROM Events ORDER BY price DESC LIMIT 1;



14. SELECT payment\_status, COUNT(\*) AS total

FROM Payments

GROUP BY payment\_status;



# Summary:

This system models a real-world booking process using relational database principles. With structured tables and optimized queries, it helps in efficient data management and reporting.

The **Event Booking System using MySQL** is a robust, relational database-driven model designed to efficiently manage users, events, bookings, and payments. The system allows users to register, browse and book events, and make payments through various methods.

Key features include:

* **User Management**: Handles registration and login of users with secure data storage.
* **Event Management**: Stores event details including title, date, time, location, and seat availability.
* **Booking System**: Links users to events with the ability to book multiple tickets per event.
* **Payment Handling**: Tracks payments made against bookings with status updates (e.g., Completed, Pending).
* **Relational Integrity**: Achieved through the use of primary and foreign keys, ensuring proper linkage and data consistency.
* **Advanced Queries**: Supports complex SQL operations like joins, aggregations, filters, and updates to offer real-time insights and administrative control.

The system is scalable, adaptable to real-world scenarios, and serves as a foundational backend for building a full-fledged event management application or website.

**Conclusion:**

The development of the **Event Booking System using MySQL** demonstrates the practical implementation of database management concepts in real-world applications. This project successfully integrates various components of DBMS such as data modeling, entity relationships, SQL queries, and normalization to create an efficient and scalable system.

Through structured tables and properly defined relationships, the system ensures data integrity and supports dynamic user interactions like event registration, ticket booking, and secure payment processing. It also provides valuable insights through analytical queries and reporting.

Overall, this project not only fulfills the academic objectives of learning MySQL and relational databases but also lays a strong foundation for building advanced web or mobile-based event management platforms in the future.