DATABASE MANAGEMENT SYSTEM PROJECT REPORT

SUBMITTED BY: ARJITA YADAV
ROLL: ACE080BCT015

Section: A

Event Management System

1. Introduction

The Event Management System is a database application designed to manage events, venues, users, bookings, and payments efficiently. The system allows users to book tickets for events, events to be organized at specific venues, and payments to be recorded for each booking. This project demonstrates database design concepts including Entity-Relationship modeling, Primary and Foreign Keys, SQL queries, aggregate functions, views, and transactions.

2. Objectives

• To design a relational database with proper relationships.

• To implement ER diagram with cardinalities.

• To demonstrate SQL operations (JOIN, GROUP BY, etc.).

• To implement view and transaction control.

• To ensure proper referential integrity using foreign keys.

3. Domain Description

The system manages multiple users who book event tickets, multiple events organized at venues, each booking linked to a user and an event, and each payment linked to a booking. This domain involves multiple entities and relationships, making it suitable for DBMS implementation.

QUERIES:

INNER JOIN:

which user booked which event:

Results Messages

	name	event_name	tickets
1	Ram Sharma	Music Concert	2
2	Sita Rai	Music Concert	1
3	Hari Thapa	Tech Seminar	3
4	Mina Gurung	Comedy Show	2
5	Bikash Karki	Tech Seminar	1
6	Aayush Adhikari	Startup Meetup	2
7	Nisha Shrestha	Dance Festival	4
8	Prakash Lama	Art Exhibition	1
9	Ram Sharma	Dance Festival	3
10	Sita Rai	Startup Meetup	2
11	Ram Sharma	Tech Seminar	2

LEFT JOIN:

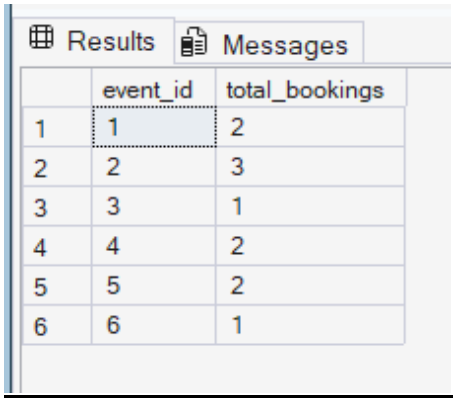
Show all events even if no booking exists:

Results Messages

	event_name	booking_id
1	Music Concert	1
2	Music Concert	2
3	Tech Seminar	3
4	Tech Seminar	5
5	Tech Seminar	12
6	Comedy Show	4
7	Startup Meetup	6
8	Startup Meetup	10
9	Dance Festival	7
10	Dance Festival	9
11	Art Exhibition	8

COUNT + GROUP BY:

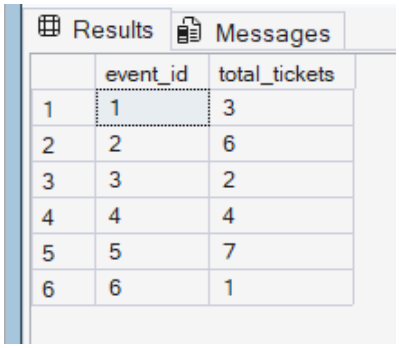
Count total bookings per event:



Results Messages

	event_id	total_bookings
1	1	2
2	2	3
3	3	1
4	4	2
5	5	2
6	6	1

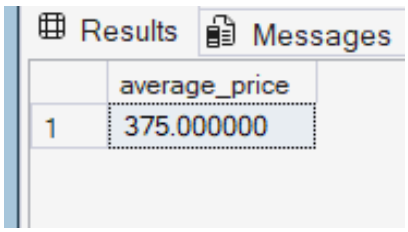
SUM [Total tickets per event]:



Results Messages

	event_id	total_tickets
1	1	3
2	2	6
3	3	2
4	4	4
5	5	7
6	6	1

AVG (Average ticket price):

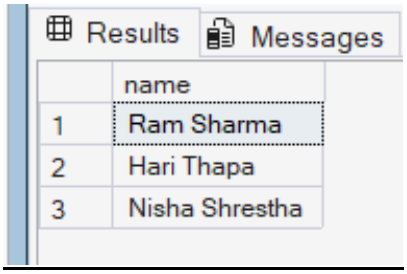


Results Messages

	average_price
1	375.000000

SUBQUERY

Users who booked more than 2 tickets:

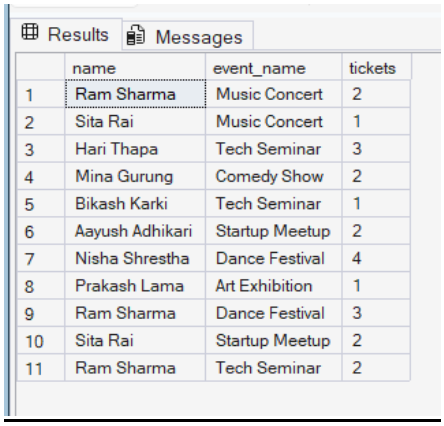


Results | Messages

	name
1	Ram Sharma
2	Hari Thapa
3	Nisha Shrestha

VIEW

Create a view to show booking details:



Results | Messages

	name	event_name	tickets
1	Ram Sharma	Music Concert	2
2	Sita Rai	Music Concert	1
3	Hari Thapa	Tech Seminar	3
4	Mina Gurung	Comedy Show	2
5	Bikash Karki	Tech Seminar	1
6	Aayush Adhikari	Startup Meetup	2
7	Nisha Shrestha	Dance Festival	4
8	Prakash Lama	Art Exhibition	1
9	Ram Sharma	Dance Festival	3
10	Sita Rai	Startup Meetup	2
11	Ram Sharma	Tech Seminar	2

4. Entity List

• Users

• Venue

• Event

• Booking

• Payment

5. ER Diagram Explanation

• Venue — hosts — Event (1:N)

• User — makes — Booking (1:N)

• Event — has — Booking (1:N)

• Booking — paid_by — Payment (1:1)

6. Relational Schema (With Data Types)

• **USERS**: user_id (PK), name VARCHAR(50), email VARCHAR(50), phone VARCHAR(15), created_at DATETIME

• **VENUE**: venue_id (PK), venue_name VARCHAR(50), city VARCHAR(50), capacity INT

• **EVENT**: event_id (PK), event_name VARCHAR(50), event_date DATE, ticket_price DECIMAL(8,2), venue_id (FK)

• **BOOKING:** booking_id (PK), user_id (FK), event_id (FK), tickets INT, booking_status VARCHAR(20)

• **PAYMENT**: payment_id (PK), booking_id (FK), amount DECIMAL(8,2), payment_method VARCHAR(20)

7. SQL Implementation

• Tables created using PRIMARY KEY and FOREIGN KEY constraints.

• Sample data inserted successfully (Users, Venue, Event, Booking, Payment).

8. Required SQL Queries

• INNER JOIN – Displays user name, event name, and tickets booked.

• LEFT JOIN – Displays all events including those without bookings.

• COUNT with GROUP BY – Counts total bookings per event.

• SUM with GROUP BY – Calculates total tickets sold per event.

• AVG – Calculates average ticket price.

• Subquery – Displays users who booked more than 2 tickets.

9. View Implementation

A view named 'BookingDetails' was created to simplify repeated query execution.

10. Transaction Control

Transaction control implemented using BEGIN TRANSACTION, COMMIT, and ROLLBACK to maintain data consistency.

11. Normalization

The database is designed up to Third Normal Form (3NF). There are no repeating groups, no partial dependency, and no transitive dependency.

12. Final Table

Entity	Primary Key	Foreign Key	Relationship
Users	user_id	-	Makes Booking (1:N)
Venue	venue_id	-	Hosts Event (1:N)
Event	event_id	venue_id	Has Booking (1:N)
Booking	booking_id	user_id, event_id	Paid by Payment (1:1)
Payment	payment_id	booking_id	Linked to Booking

13. Conclusion

The Event Management System successfully demonstrates proper database design, Entity-Relationship modeling, use of primary and foreign keys, implementation of SQL queries, aggregate functions, view creation, and transaction control while maintaining referential integrity.

Github link: https://github.com/Arjita15/Database_project/tree/main