Database Management System (UEC716)

Event Reservation System

3O1D

Sakshi Rana 102215293
Rahul Gupta 102215298
Mrikulesh Minhas 102215357

Submitted to
Dr. Ashu Sharma

DEPARTMENT OF ELECTRONICS and COMMUNICATION
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,
(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB
INDIA
Aug-Dec 2024

ACKNOWLEDGEMENT

It is with immense pleasure that we present the report of our DBMS Project undertaken during the 5th semester. We are deeply grateful to our esteemed mentor, Dr. Ashu Sharma, Professor in the Department of Electronics and Communication at Thapar University, Patiala, for her unwavering support and guidance throughout our journey. Her dedication, thoroughness, and perseverance have served as a constant source of inspiration for us.

We would also like to extend our heartfelt appreciation to our group members and seniors at the college for their invaluable advice, continuous guidance, and unwavering encouragement throughout the preparation of this project. Their insights and support have truly enriched our learning experience.

Lastly, we express our sincere gratitude to the college management for fostering an environment conducive to our academic pursuits. Their support has been instrumental in enabling us to undertake and complete this project successfully.

ABSTRACT

The project presents the development of the Event Reservation System aimed at streamlining the organisation and management of events across various venues. The system handles key functionalities such as event scheduling, venue allocation, ticket reservations, audience feedback collection, and revenue tracking.
Using the relational database management system (RDBMS) MySQL, the system is designed to manage entities like events, venues, audience, and organisers, with efficient data relationships to support quick information retrieval.

In conclusion, the Event Reservation System provides an efficient, scalable solution for organising and managing events, benefiting both organisers and attendees by improving operational workflows and enhancing the overall event experience.

SYSTEM OVERVIEW

Our Event Reservation System is an integrated platform designed to streamline the organisation, booking, and management of events. It focuses on managing a variety of entities essential for event operations, including Events, Venues, Audience, Organizers, Revenue, Tickets, and Feedback. By establishing clear relationships between these entities, the system ensures a seamless experience for users while maintaining an organised data structure for efficient event handling.

Key Features of the System

Comprehensive Event Management: The system allows for the easy management of events, with features that enable adding, updating, and retrieving essential event details such as event types, dates, and times. It ensures that each event is properly managed and associated with relevant venues and organisers.

Venue and Audience Coordination: Venues are linked to events, and each venue's details, such as capacity, location, and cost, are efficiently managed. The system also tracks audience members who book tickets, storing their personal details such as name, age, contact information, and event participation. This ensures that the venue capacity is never exceeded and that event organisers have a clear understanding of their audience.

Revenue Tracking and Ticketing: The system is capable of tracking the revenue generated by each event, which provides financial insights for organisers. It also manages the ticketing process, linking ticket availability to specific events, and ensuring that each audience member's ticket purchase is accurately recorded. This functionality helps event organisers plan and assess their event's financial performance.

Feedback Collection: After events, the system enables audience members to provide feedback on their experiences. This feedback is vital for organisers as it offers insights into the event's success and highlights areas for improvement, which is essential for continuous enhancement of future events.

DATABASE REQUIREMENT

1. Event Scheduling

The system allows users to schedule different types of events with specific dates and time, providing an organized way to manage event timelines.

2. Venue Allocation and Management

The system manages detailed information about event venues, including their location, capacity, and cost. It ensures venues are appropriately allocated for events based on availability and requirements.

3. Organiser Coordination

The system manages the details of event organisers including their contact, address and email, enabling efficient communication.

4. Audience Registration

The system maintains records of attendees for each event, providing a comprehensive view of the audience and ensuring streamlined ticket allocation.

5. Revenue Monitoring

The system monitors the total revenue generated from each event.

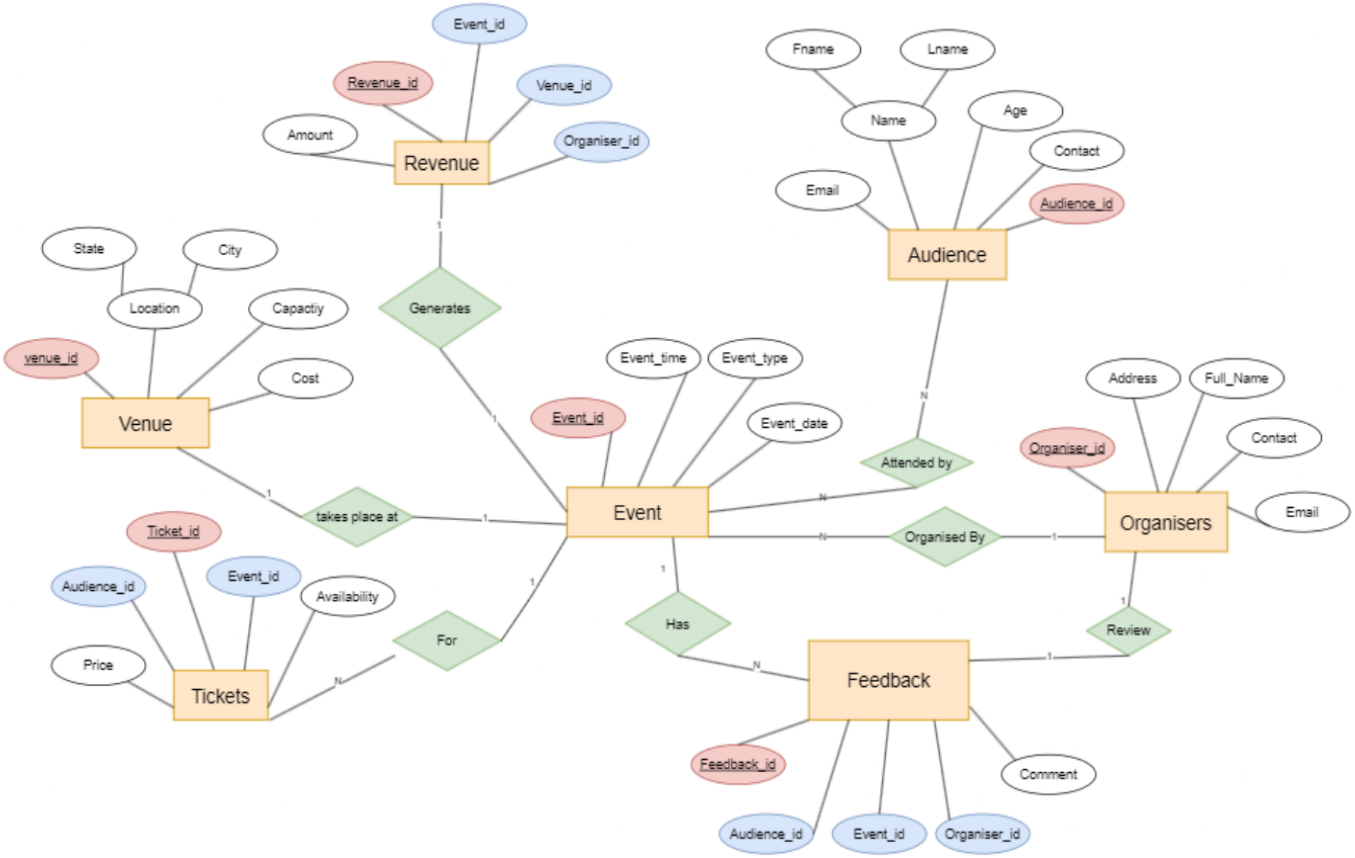
6.Ticket Availability Monitoring

The system records the availability of tickets for different event types along with their prices.

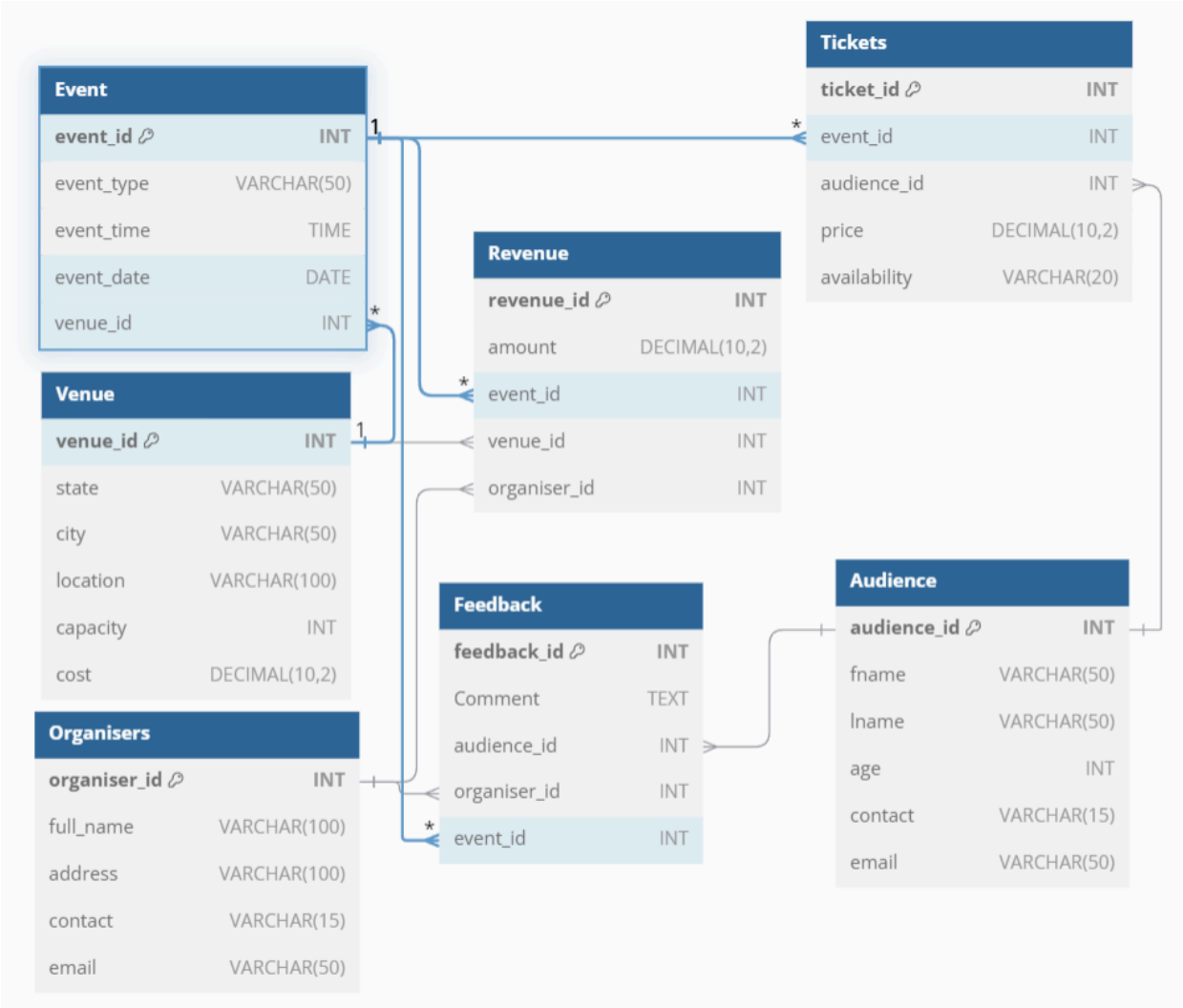
7. Feedback Analysis

The system records feedback from both the audience and organizers, helping improve future events by analyzing strengths and areas for improvement.

ER DIAGRAM


Event_id
Revenue_id
Venue_id
Amount
Organiser_id
Revenue
Fname
Lname
Name
Age
Contact
Email
Audience_id
Audience
State
City
Location
Capactiy
Generates
venue_id
Cost
Venue
Event_time
Event_type
Event_id
Event_date
Address
Full_Name
Organiser_id
Contact
Attended by
takes place at
Event
Organised By
Organisers
Email
Ticket_id
Audience_id
Event_id
Availability
Price
Tickets
For
Has
Review
Feedback
Feedback_id
Comment
Audience_id
Event_id
Organiser_id

RELATIONAL SCHEMA



ER to Table

Venue Table

Venue Table

Column	Null?	Type
venue_id	NOT NULL	INT AUTO_INCREMENT
state	NULL	VARCHAR(50)
city	NULL	VARCHAR(50)
location	NULL	VARCHAR(100)
capacity	NULL	INT
cost	NULL	DECIMAL(10,2)

Event Table

Event Table

Column	Null?	Type
event_id	NOT NULL	INT AUTO_INCREMENT
event_type	NULL	VARCHAR(50)
event_datetime	NULL	DATETIME
venue_id	NULL	INT

Audience Table

Audience Table

Column	Null?	Type
audience_id	NOT NULL	INT AUTO_INCREMENT
fname	NULL	VARCHAR(50)
lname	NULL	VARCHAR(50)
age	NULL	INT
contact	NULL	VARCHAR(15)
email	NULL	VARCHAR(50) UNIQUE

Organisers Table

Organisers Table

Column	Null?	Type
organiser_id	NOT NULL	INT AUTO_INCREMENT
full_name	NULL	VARCHAR(100)
address	NULL	VARCHAR(100)
contact	NULL	VARCHAR(15)
email	NULL	VARCHAR(50) UNIQUE

Feedback Table

Feedback Table

Column	Null?	Type
feedback_id	NOT NULL	INT AUTO_INCREMENT
comment	NULL	TEXT
rating	NULL	TINYINT CHECK (rating BETWEEN 1 AND 5)
event_id	NULL	INT
audience_id	NULL	INT
organiser_id	NULL	INT

Revenue Table

Revenue Table

Column	Null?	Type
revenue_id	NOT NULL	INT AUTO_INCREMENT
amount	NULL	DECIMAL(10,2)
event_id	NULL	INT
venue_id	NULL	INT
organiser_id	NULL	INT

Tickets Table

Tickets Table

Column	Null?	Type
ticket_id	NOT NULL	INT AUTO_INCREMENT
event_id	NULL	INT
audience_id	NULL	INT
price	NULL	DECIMAL(10,2)
availability	NULL	ENUM('available','sold','reserved')

NORMALISATION

1. **First Normal Form (1NF):**

- **Criteria: All attributes must contain atomic (indivisible) values, and each value must be stored in a unique cell.**
- **Status in ERS:**
  - ❖ **Organisers Table violated 1NF because it had multiple values of address and contact for 3 organisers.**
  - ❖ **For 1NF normalisation two tables were created:OrganiserAddresses and OrganiserContacts.**

```
ALTER TABLE Organisers
MODIFY COLUMN contact VARCHAR(200);

ALTER TABLE Organisers
MODIFY COLUMN address VARCHAR(200);

UPDATE Organisers
SET address = 'Address 1; Address 2',
    contact = '1234567890, 9876543210'
WHERE organiser_id = 5;

UPDATE Organisers
SET address = 'Address 1; Address 2',
    contact = '6783670197, 960368650'
WHERE organiser_id = 10;
```

```
UPDATE Organisers
SET address = 'Address 1; Address 2',
    contact = '7563945694, 7258430385'
WHERE organiser_id = 1;

CREATE TABLE OrganiserAddresses (
    organiser_id INT,
    address VARCHAR(100),
    FOREIGN KEY (organiser_id) REFERENCES Organisers(organiser_id)
);

CREATE TABLE OrganiserContacts (
    organiser_id INT,
    contact VARCHAR(15),
    FOREIGN KEY (organiser_id) REFERENCES Organisers(organiser_id)
);
```

```
INSERT INTO OrganiserAddresses (organiser_id, address)
SELECT organiser_id, 'Address1' FROM Organisers WHERE organiser_id = 1;

INSERT INTO OrganiserAddresses (organiser_id, address)
SELECT organiser_id, 'Address2' FROM Organisers WHERE organiser_id = 1;

INSERT INTO OrganiserAddresses (organiser_id, address)
SELECT organiser_id, 'Address1' FROM Organisers WHERE organiser_id = 10;

INSERT INTO OrganiserAddresses (organiser_id, address)
SELECT organiser_id, 'Address2' FROM Organisers WHERE organiser_id = 10;

INSERT INTO OrganiserAddresses (organiser_id, address)
SELECT organiser_id, 'Address1' FROM Organisers WHERE organiser_id = 5;

INSERT INTO OrganiserAddresses (organiser_id, address)
SELECT organiser_id, 'Address2' FROM Organisers WHERE organiser_id = 5;
```

```
INSERT INTO OrganiserContacts (organiser_id, contact)
SELECT organiser_id, '1234567890' FROM Organisers WHERE organiser_id = 1;

INSERT INTO OrganiserContacts (organiser_id, contact)
SELECT organiser_id, '9876543210' FROM Organisers WHERE organiser_id = 1;
```

After Normalisation:

Organisers Table

organiser_id	full_name	email
1	Alice Cooper	alice.cooper@google.com
2	Michael Brown	michael.brown@google.com
3	David Lee	david.lee@google.com
4	Sandra White	sandra.white@google.com
5	Lisa Green	lisa.green@google.com
6	John King	john.king@google.com
7	Betty Adams	betty.adams@google.com
8	Charles Harris	charles.harris@google.com
9	Rachel Young	rachel.young@google.com
10	William White	william.white@google.com

OrganiserAddresses Table

organiser_id	address
1	Address1
1	Address2
10	Address1
10	Address2
5	Address1
5	Address2

OrganiserContacts Table

organiser_id	contact
1	1234567890
1	9876543210
10	6783670197
10	960368650
5	1234567890
5	9876543210

2. **Second Normal Form (2NF):**

- **Criteria: Achieved when the database is in 1NF, and all non-key attributes are fully functionally dependent on the primary key.**
- **Status in ERS:**
  - **Each table has a clearly defined primary key (e.g., `Event_id`, `Audience_id`, `Venue_id`).**
  - **For composite keys (e.g., `Ticket_id` linking `Event_id` and `Audience_id`), no partial dependencies exist.**
  - **Covered.**

3. **Third Normal Form (3NF):**

- **Criteria: Achieved when the database is in 2NF, and all attributes are only dependent on the primary key, not on other non-key attributes.**
- **Status in ERS:**
    - **Example: Attributes like `State`, `City`, and `Location` in the `Venue` entity are directly dependent on `Venue_id` (primary key).**
    - **There are no transitive dependencies indicated in the description.**
    - **Covered.**

4. **Boyce-Codd Normal Form (BCNF):**

- **Criteria: A stricter version of 3NF where every determinant is a candidate key.**
- **Status in ERS:**
    - **Relationships and dependencies (e.g., `Organiser_id` determining events in the `Organisers` table) seem consistent with BCNF standards.**
    - **Likely covered.**

5. **Fourth Normal Form (4NF):**

- **Criteria: No multivalued dependencies exist unless they are trivial.**
- **Status in ERS:**
    - **The ERD handles many-to-many relationships with associative entities, avoiding multivalued dependencies.**

    - **Covered.**

6. **Fifth Normal Form (5NF):**

- **Criteria: Decomposed tables should be reconstructed without loss of data and redundancy, even for complex join dependencies.**
- **Status in ERS:**
    - **Relationships like `Attended by` (Audience-Event) and `Tickets` appear normalized to avoid redundant data.**
    - **Likely covered.**

Queries and their Respective Outputs

1.Revenue Monitoring:

```
-- Find the total revenue generated by each event
SELECT E.event_type, SUM(R.amount) AS total_revenue
FROM Revenue R
JOIN Event E ON R.event_id = E.event_id
GROUP BY E.event_type;
```

Output:

event_type	total_revenue
Concert	5000.00
Wedding	15700.00
Conference	3200.00
Trade Show	4000.00
Festival	6000.00
Exhibition	5800.00
Seminar	7100.00
Workshop	3600.00
Product Launch	3000.00
Birthday Party	4500.00

2. Audience Registration:

```
-- Retrieve the list of all attendees for a specific event
SELECT A.fname, A.lname
FROM Tickets T
JOIN Audience A ON T.audience_id = A.audience_id
WHERE T.event_id = 1;
```

Output:

	fname	lname
▶	John	Doe
	Jane	Smith
	Chris	Johnson

3. Ticket Availability:

```sql
-- Retrieve Events with Total Ticket Sales Above a Certain Threshold
SELECT E.event_type, SUM(T.price) AS total_sales
FROM Tickets T
JOIN Event E ON T.event_id = E.event_id
GROUP BY E.event_type
HAVING SUM(T.price) > 50;
```

Output:

	event_type	total_sales
▶	Concert	150.00
	Wedding	300.00
	Conference	225.00
	Trade Show	90.00
	Festival	450.00
	Exhibition	360.00
	Seminar	120.00

4.Feedback Analysis:

```sql
-- Find Events with Feedback from Attendees Over Age 30
SELECT E.event_type, A.fname, A.lname, F.comment
FROM Feedback F
JOIN Audience A ON F.audience_id = A.audience_id
JOIN Event E ON F.event_id = E.event_id
WHERE A.age > 20;
```

Output:

event_type	fname	lname	comment
Concert	Chris	Johnson	Not bad, but could be better organized.
Wedding	Patricia	Williams	The wedding was beautiful, very well organized!
Wedding	Linda	Jones	Loved the decorations and atmosphere!
Conference	David	Garcia	Great conference, very informative sessions!
Conference	Susan	Miller	The sessions were a bit too long, but overall go...
Conference	James	Martinez	Fantastic conference, learned a lot!
Trade Show	Robert	Lopez	Loved the product displays, very interesting!
Trade Show	Jennifer	Gonzalez	The trade show was okay, but could be more in...
Festival	Elizabeth	Wilson	The festival was great, but the food stalls were...
Festival	Joshua	Anderson	Great vibe, excellent performances!

5. Venue Allocation and Management:

```
-- Count Events by Venue with Average Cost and Total Capacity
SELECT V.state, V.city, COUNT(E.event_id) AS total_events, AVG(V.cost) AS avg_cost,
SUM(V.capacity) AS total_capacity
FROM Venue V
LEFT JOIN Event E ON V.venue_id = E.venue_id
GROUP BY V.state, V.city;
```

Output:

state	city	total_events	avg_cost	total_capacity
California	Los Angeles	1	10000.000000	5000
New York	New York City	1	15000.000000	3000
Texas	Austin	1	12000.000000	7000
Florida	Miami	1	11000.000000	4000
Nevada	Las Vegas	1	13000.000000	6000
Illinois	Chicago	1	20000.000000	10000
Ohio	Cleveland	1	9500.000000	3500
Michigan	Detroit	1	14000.000000	8000
Arizona	Phoenix	1	12000.000000	5000
Colorado	Denver	1	11500.000000	4500

6.Organiser Coordination:

```
412     -- List Events with Multiple Organisers
413 •   SELECT E.event_type, COUNT(O.organiser_id) AS organiser_count
414     FROM Event E
415     JOIN Feedback F ON E.event_id = F.event_id
416     JOIN Organisers O ON F.organiser_id = O.organiser_id
417     GROUP BY E.event_type
418     HAVING COUNT(O.organiser_id) > 1;
```

Output:

event_type	organiser_count
Concert	3
Wedding	3
Conference	3
Trade Show	3
Festival	3
Exhibition	3
Seminar	2

7.Event Scheduling:

```
    -- Retrieve Events with Audience Count
•   SELECT E.event_id, E.event_type, COUNT(T.ticket_id) AS audience_count
    FROM Event E
    LEFT JOIN Tickets T ON E.event_id = T.event_id
    GROUP BY E.event_id, E.event_type;
```

Output:

	event_id	event_type	audience_count
▶	1	Concert	3
	2	Wedding	3
	3	Conference	3
	4	Trade Show	3
	5	Festival	3
	6	Exhibition	3
	7	Seminar	2
	8	Workshop	0
	9	Wedding	0
	10	Product Launch	0

8..Feedback Analysis:

```
-- Find Feedback Given by a Specific Audience
SELECT F.feedback_id, F.comment, E.event_type, A.fname, A.lname
FROM Feedback F
JOIN Audience A ON F.audience_id = A.audience_id
JOIN Event E ON F.event_id = E.event_id
WHERE A.audience_id = 1;
```

	feedback_id	comment	event_type	fname	lname
▶	1	Great concert, would love to attend again!	Concert	John	Doe

9.Ticket Availability:

```sql
-- Find the Average Ticket Price for Each Event Type
SELECT E.event_type, AVG(T.price) AS average_ticket_price
FROM Tickets T
JOIN Event E ON T.event_id = E.event_id
GROUP BY E.event_type;
```

event_type	average_ticket_price
Concert	50.000000
Wedding	100.000000
Conference	75.000000
Trade Show	30.000000
Festival	150.000000
Exhibition	120.000000
Seminar	60.000000

10.Ticket Availability:

```sql
-- Retrieve the Number of Available Tickets for Each Event
SELECT E.event_id, E.event_type, COUNT(T.ticket_id) AS available_tickets
FROM Tickets T
JOIN Event E ON T.event_id = E.event_id
WHERE T.availability = 'Available'
GROUP BY E.event_id, E.event_type;
```

event_id	event_type	available_tickets
1	Concert	2
2	Wedding	2
3	Conference	2
4	Trade Show	2
5	Festival	2
6	Exhibition	2
7	Seminar	1

11.Feedback Analysis:

```
-- Find Events with Missing Feedback
SELECT E.event_id, E.event_type, E.event_date
FROM Event E
LEFT JOIN Feedback F ON E.event_id = F.event_id
WHERE F.feedback_id IS NULL;
```

	event_id	event_type	event_date
▶	8	Workshop	2025-04-05
	9	Wedding	2025-05-10
	10	Product Launch	2025-06-15
	11	Birthday Party	2025-07-20
	12	Charity Event	2025-08-01
	13	Networking	2025-09-10
	14	Job Fair	2025-10-22
	15	Art Show	2025-11-18
	16	Health Expo	2025-12-14
	17	Food Festival	2025-01-28

12.Ticket Availability:

```
--  Find Total Ticket Sales by Event Type
SELECT E.event_type, SUM(T.price) AS total_ticket_sales
FROM Tickets T
JOIN Event E ON T.event_id = E.event_id
GROUP BY E.event_type;
```

	event_type	total_ticket_sales
▶	Concert	150.00
	Wedding	300.00
	Conference	225.00
	Trade Show	90.00
	Festival	450.00
	Exhibition	360.00
	Seminar	120.00

13.Feedback Analysis:

```
-- Identify Events with Feedback Keywords (e.g., "Amazing" or "Better")
SELECT E.event_type, F.comment
FROM Feedback F
JOIN Event E ON F.event_id = E.event_id
WHERE F.comment LIKE '%Amazing%' OR F.comment LIKE '%Better%';
```

event_type	comment
Concert	Amazing experience, well organized.
Concert	Not bad, but could be better organized.
Festival	Amazing music festival! Loved every performance!

14. Audience Registration:

```
-- Find Average Age of Attendees for Each Event
SELECT E.event_type, AVG(A.age) AS avg_age
FROM Tickets T
JOIN Audience A ON T.audience_id = A.audience_id
JOIN Event E ON T.event_id = E.event_id
GROUP BY E.event_type;
```

event_type	avg_age
Concert	30.0000
Wedding	33.3333
Conference	44.3333
Trade Show	47.0000
Festival	33.3333
Exhibition	36.6667
Seminar	37.5000

CONCLUSION:

The Event Reservation System efficiently organizes and tracks events, venues, audiences, and revenue, providing a seamless experience for event organizers. With features like event scheduling, venue allocation, ticket availability monitoring, and feedback analysis, it offers a structured approach to managing various aspects of event planning. This project emphasizes the importance of a well-designed database, ensuring accurate storage, retrieval, and processing of event-related data, ultimately simplifying the complexities of organizing and executing successful events.