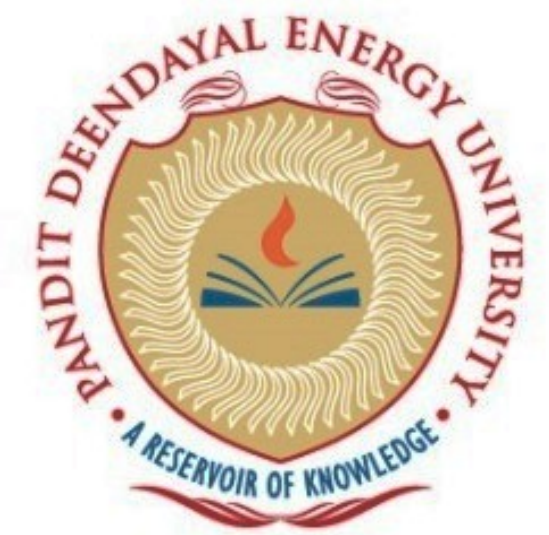
PANDIT DEENDAYAL ENERGY UNIVERSITY

DATABASE MANAGEMENT LAB PROJECT

(SQL)

EVENT TICKET MANAGEMENT SYSTEM



Submitted to

Nandini Modi

Table of Contents

Overview of Project

Our Ticket Management System is a comprehensive platform designed to streamline the process of booking tickets for various events and entertainment shows. Much like popular ticketing platforms such as BookMyShow, our system offers a user-friendly interface that enables users to effortlessly browse, select, and purchase tickets for a wide range of events.

Key Features:

Event Variety: Our platform caters to diverse interests by offering tickets for a variety of events, including movie screenings, concerts, art exhibitions, sports matches, and stand-up comedy shows.

a.)Easy Booking Process: Users can conveniently browse through available events, select their preferred show, choose their desired seats, and specify timings all within a few clicks.

b.)Seat Selection: The system provides interactive seat maps for venues, allowing users to visually select their seats based on their preferences, such as proximity to the stage or screen.

c.)Secure Transactions: With robust payment gateways integrated into the system, users can rest assured that their transactions are secure and protected.

d.)Personalized Experience: Users can create accounts on the platform to access personalized recommendations, view their booking history, and receive notifications about upcoming events matching their interests.

e.)Real-time Availability: The system continuously updates event availability in real-time, ensuring that users have access to the latest information regarding show timings, seat availability, and ticket prices.

Services Provided:

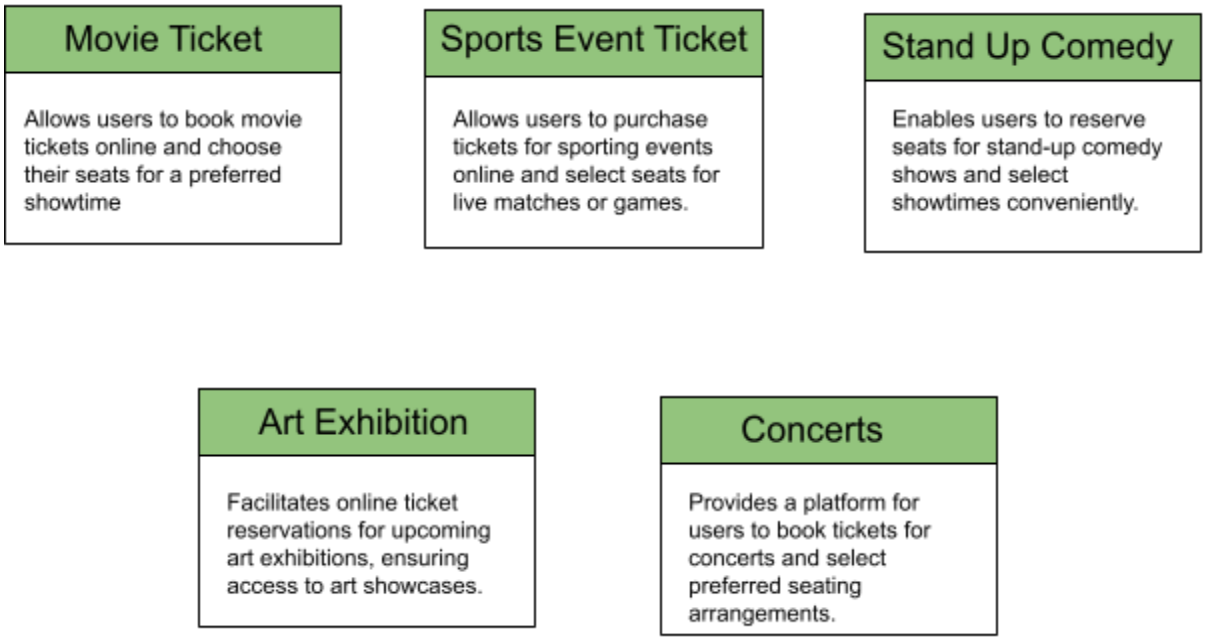
a.)Movie Ticket Booking

b.)Stand-Up Comedy Ticket Booking:

c.)Art Exhibition Ticket Booking:

d.)Concert Ticket Booking:

e.)Sports Ticket Booking:



Facilitating with UI design and Prototype with the help of Figma Software.

ER Diagram

STEP 1: SELECTION OF ENTITIES

1.)Admin:

Entity Description: Stores information about administrators who manage the entire database.

2.)Customer_info:

Entity Description: Contains details of all customers who have signed in to the platform.

3.)Event_list:

Entity Description: Provides information about events conducted by the organization, including event names, dates, and other relevant details.

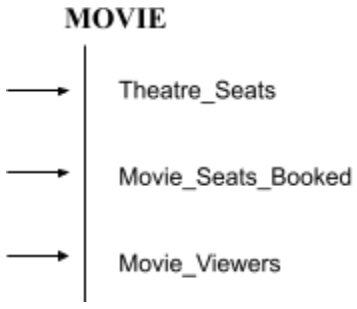
4.)Venue_list:

Entity Description: Stores specific locations where the firm conducts events, including venue names and addresses.

5.)Events:

a.)Movie:

Entity Description: Manages movie-related information such as available seats in theaters, seats booked by users, and details of movie viewers.

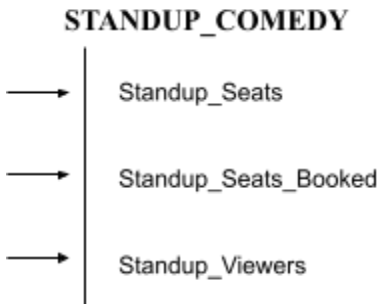


b.)Concert:

Entity Description: Records details of concerts including venue, event name, and other relevant information. Tracks concert attendees.

c.)Stand-Up Comedy:

Entity Description: Manages seat information for stand-up comedy events, including seats available and seats booked. Also tracks viewers of stand-up comedy shows.

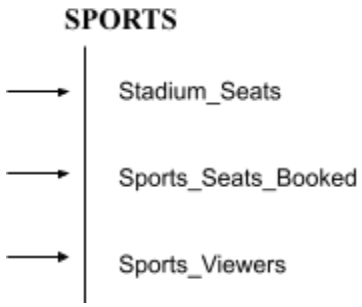


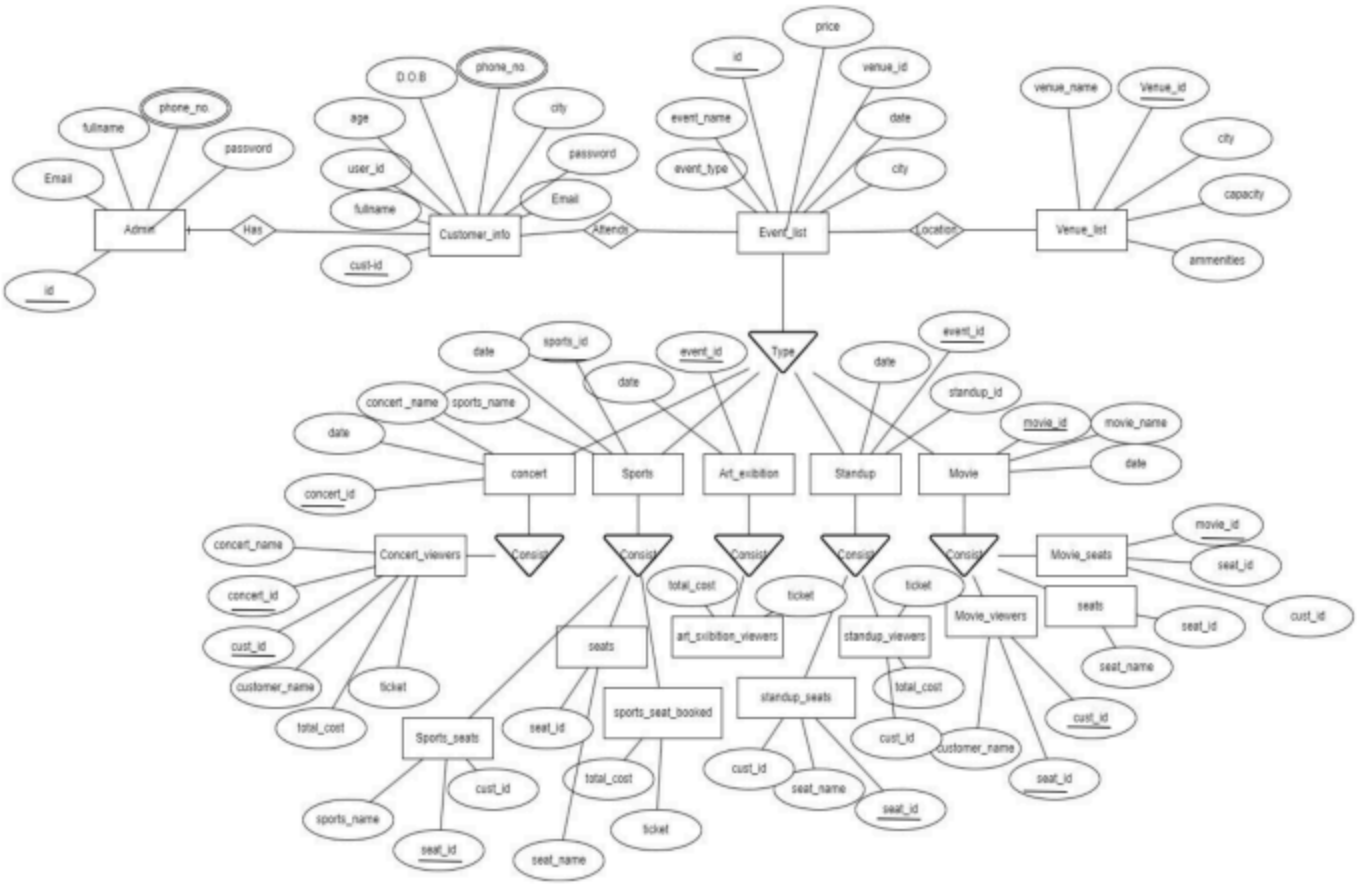
d.)Art Exhibition:

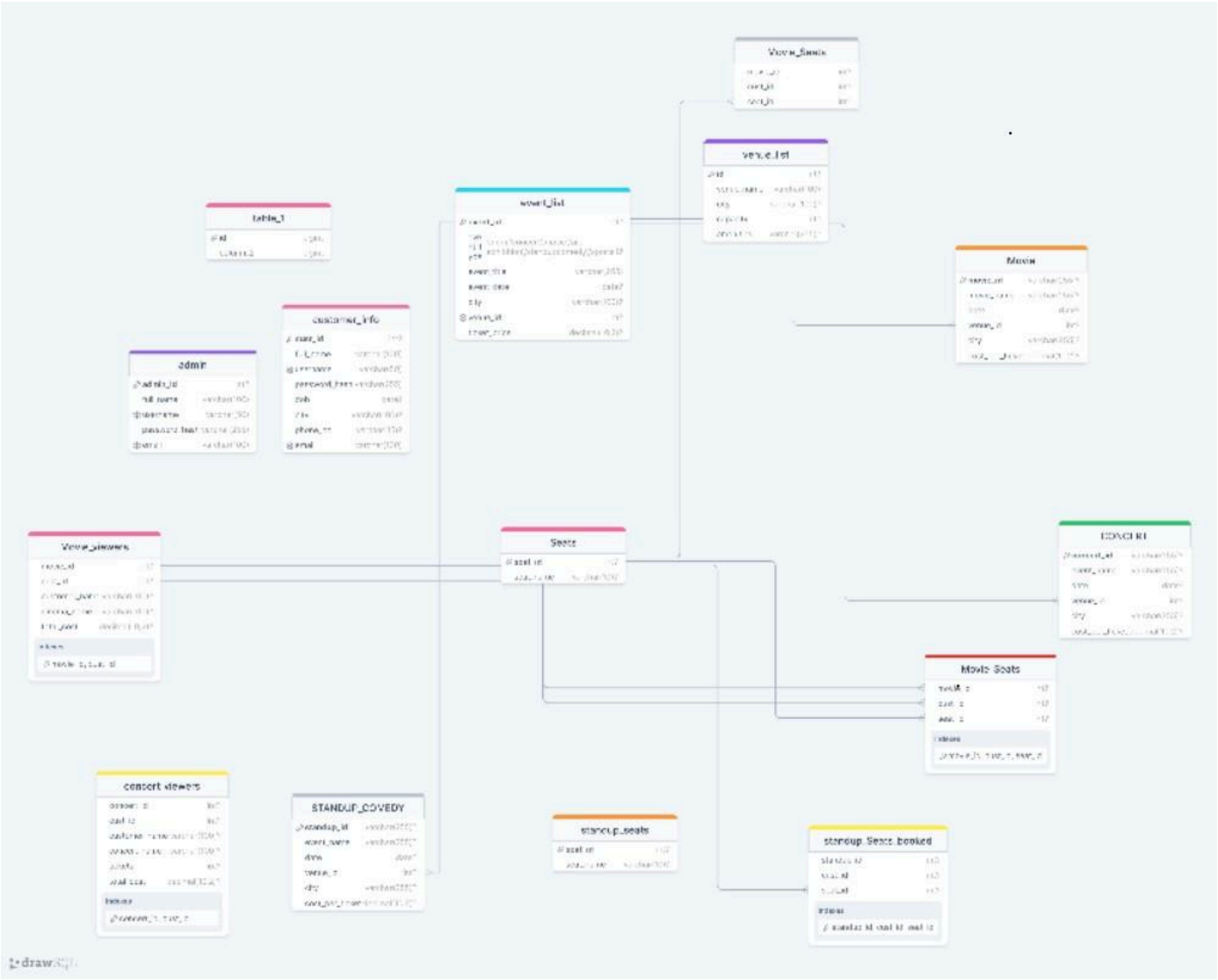
Entity Description: Stores information about art exhibitions, including venue details and exhibition names. Tracks viewers of art exhibitions.

e.)Sports:

Entity Description: Manages stadium seat information for sports events, including available seats and seats booked. Also tracks viewers of sports events.




Admin
phone_no
fullname
password
Email
id
Has
Customer_info
D.O.B
phone_no
age
city
user_id
password
fullname
Email
cust-id
Attend
Event_list
price
id
venue_id
event_name
date
event_type
city
Location
Venue_list
venue_name
Venue_id
city
capacity
ammenities
Type
concert
Sports
Art_exibition
Standup
Movie
concert_id
concert_name
sports_name
date
sports_id
event_id
standup_id
movie_id
movie_name
Consist
Concert_viewers
concert_name
concert_id
cust_id
customer_name
ticket
total_cost
seats
Sports_seats
seat_id
cust_id
sports_name
seat_id
sports_seat_booked
total_cost
seat_name
art_exibition_viewers
standup_viewers
standup_seats
Movie_viewers
Movie_seats
movie_id
seat_id
cust_id
seat_name


Venue_Seats
venue_list
table_1
event_list
Movie
customer_info
admin
Venue_viewers
Seats
Movie Seats
concert viewers
STANDUP_COMEDY
standup_seats
standup Seats booked

ER Diagram to Relational Model

---

1. Admin Table:
   - Represents administrators managing the database.
   - Attributes include `admin_id`, `full_name`, `username`, `password_hash`, and `email`.
   - `admin_id` is the primary key, ensuring each administrator has a unique identifier.
   - `username` and `email` are marked as `UNIQUE` to prevent duplication.

2. Customer_info Table:
   - Stores information about customers who have signed up.
   - Attributes include `cust_id`, `full_name`, `username`, `password_hash`, `dob`, `city`, `phone_no`, and `email`.
   - `cust_id` serves as the primary key for identifying customers uniquely.
   - `username` and `email` are marked as `UNIQUE` to ensure uniqueness.

3. Event_list Table:
   - Contains details of events conducted by the organization.
   - Attributes include `event_id`, `event_type`, `event_title`, `event_date`, `city`, `venue_id`, and `ticket_price`.
   - `event_id` serves as the primary key to uniquely identify each event.
   - `venue_id` is a foreign key referencing the `venue_list` table, establishing a relationship between events and venues.

4. Venue_list Table:
   - Stores specific locations where events are conducted.
   - Attributes include `venue_id`, `venue_name`, `city`, `capacity`, and `amenities`.
   - `venue_id` serves as the primary key to uniquely identify each venue.
   - `venue_id` in the `event_list` table acts as a foreign key, linking events to venues.

5. Events Tables: (Movie, Concert, Standup_Comedy, Art_Exhibition, Sports):
   - Each event type has its own table storing specific details.
   - Tables include attributes such as `event_id`, `event_name`, `date`, `venue_id`, `city`, and `cost_per_ticket`.
   - `event_id` serves as the primary key in each table.
   - `venue_id` acts as a foreign key referencing the `venue_list` table, linking events to venues.

6. Additional Tables Under Each Event Type:
   - Each event type (Movie, Concert, Standup_Comedy, Art_Exhibition, Sports) has additional tables specific to its functionality.
   - These tables include attributes for seats, bookings, and viewers related to each event type.
   - For example, the `Movie` table has associated tables `Seats`, `Movie_Seats`, and `Movie_viewers`.

Table Name	Primary Key	Foreign Key
Admin	admin_id	-
Customer_info	cust_id	-
Event_list	event_id	venue_id (foreign key referencing venue_list)
Venue_list	venue_id	-
Movie	movie_id	venue_id (foreign key referencing event_list)
Concert	concert_id	venue_id (foreign key referencing event_list)
Standup_Comedy	standup_id	venue_id (foreign key referencing event_list)
Art_Exhibition	exhibition_id	venue_id (foreign key referencing event_list)
Sports	sports_id	venue_id (foreign key referencing event_list)
Seats	seat_id	-
Movie_Seats	(composite primary key)	movie_id, cust_id, seat_id (foreign keys referencing Movie and Seats tables)

Movie_viewers	(composite primary key)	movie_id, cust_id (foreign keys referencing Movie and Customer_info tables)
Concert_viewers	(composite primary key)	concert_id, cust_id (foreign keys referencing Concert and Customer_info tables)
Standup_viewers	(composite primary key)	standup_id, cust_id (foreign keys referencing Standup_Comedy and Customer_info tables)
Exhibition_viewers	(composite primary key)	exhibition_id, cust_id (foreign keys referencing Art_Exhibition and Customer_info tables)
Sports_viewers	(composite primary key)	sports_id, cust_id (foreign keys referencing Sports and Customer_info tables)

TABLES INVOLVED IN JOINS:

Tables Involved	Purpose of Join	Type of Join
Admin, Event_list	Retrieve events managed by specific admins	LEFT JOIN
Customer_info, Movie_viewers	Retrieve movie viewers along with customer info	INNER JOIN
Customer_info, Concert_viewers	Retrieve concert viewers along with customer info	INNER JOIN
Customer_info, Standup_viewers	Retrieve stand-up comedy viewers along with customer info	INNER JOIN
Customer_info, Exhibition_viewers	Retrieve art exhibition viewers along with customer info	INNER JOIN
Customer_info, Sports_viewers	Retrieve sports event viewers along with customer info	INNER JOIN
Event_list, Venue_list	Retrieve event details along with venue information	INNER JOIN
Event_list, Movie	Retrieve movie events along with event details	INNER JOIN
Event_list, Concert	Retrieve concert events along with event details	INNER JOIN
Event_list, Standup_Comedy	Retrieve stand-up comedy events along with event details	INNER JOIN
Event_list, Art_Exhibition	Retrieve art exhibition events along with event details	INNER JOIN
Event_list, Sports	Retrieve sports events along with event details	INNER JOIN
Movie, Movie_Seats	Retrieve movie events along with seat information	INNER JOIN
Movie, Movie_viewers	Retrieve movie events along with viewer information	LEFT JOIN

Concert, Concert_viewers	Retrieve concert events along with viewer information	LEFT JOIN
Standup_Comedy, Standup_viewers	Retrieve stand-up comedy events along with viewer information	LEFT JOIN
Art_Exhibition, Exhibition_viewers	Retrieve art exhibition events along with viewer information	LEFT JOIN
Sports, Sports_viewers	Retrieve sports events along with viewer information	LEFT JOIN

Normalization

FUNCTIONAL DEPENDENCIES:

1.)Admin Table (Admin):

admin_id → full_name, username, password_hash, email

username, email → admin_id (assuming username and email are unique identifiers)

2.)Customer_info Table (Customer_info):

cust_id → full_name, username, password_hash, dob, city, phone_no, email

username, email → cust_id (assuming username and email are unique identifiers)

3.)Event_list Table (Event_list):

event_id → event_type, event_title, event_date, city, venue_id, ticket_price

4.)Venue_list Table (Venue_list):

venue_id → venue_name, city, capacity, amenities

5.)Movie Table (Movie):

movie_id → movie_name, date, venue_id, city, cost_per_ticket

Seats Table (Seats):

seat_id → seat_name

Movie_Seats Table (Movie_Seats):

(movie_id, cust_id, seat_id) → {no functional dependencies}

Movie_viewers Table (Movie_viewers):

(movie_id, cust_id) → customer_name, cinema_name, total_cost

6.)Concert Table (Concert):

concert_id → event_name, date, venue_id, city, cost_per_ticket

Concert_viewers Table (Concert_viewers):

(concert_id, cust_id) → customer_name, concert_name, tickets, total_cost

7.)Standup_Comedy Table (Standup_Comedy):

standup_id → event_name, date, venue_id, city, cost_per_ticket

Standup_seats Table (Standup_seats):

seat_id → seat_name

Standup_Seats_booked Table (Standup_Seats_booked):

(standup_id, cust_id, seat_id) → {no functional dependencies}

Standup_viewers Table (Standup_viewers):

(standup_id, cust_id) → customer_name, event_name, total_cost

8.)Art_Exhibition Table (Art_Exhibition):

exhibition_id → event_name, date, venue_id, city, cost_per_ticket

Exhibition_viewers Table (Exhibition_viewers):

(exhibition_id, cust_id) → customer_name, concert_name, tickets, total_cost

9.)Sports Table (Sports):

sports_id → sport_name, date, venue_id, city, cost_per_ticket

Stadium_seats Table (Stadium_seats):

seat_id → seat_name

Sport_Seats_booked Table (Sport_Seats_booked):

(sport_id, cust_id, seat_id) → {no functional dependencies}

Sports_viewers Table (Sports_viewers):

(sport_id, cust_id) → customer_name, cinema_name, total_cost

VIOLATIONS OF NORMALIZATION NORMS:

To identify where normalization norms are violated, we need to look for the following issues:

First Normal Form (1NF)	Second Normal Form (2NF)	Third Normal Form (3NF)
Ensuring atomicity: Each column should hold only atomic values, i.e., no repeating groups or arrays.	Ensuring atomicity: Each column should hold only atomic values, i.e., no repeating groups or arrays.	Ensuring atomicity: Each column should hold only atomic values, i.e., no repeating groups or arrays.

Let's analyze each table to identify where these normalization norms may be violated:

1. Admin Table:
   - No apparent violations. All attributes are already atomic, and there are no partial or transitive dependencies.

2. Customer_info Table:
   - No apparent violations. Similar to the Admin table, all attributes are atomic, and there are no partial or transitive dependencies.

3. Event_list Table:
   - Violation of 2NF: The `event_type` attribute determines other attributes. However, this is acceptable because `event_type` is functionally dependent on the primary key (`event_id`).
   - No apparent violations of 1NF or 3NF.

4. Venue_list Table:

- No apparent violations. All attributes are atomic, and there are no partial or transitive dependencies.

5. Movie Table:
- Violation of 1NF: The `venue_id` attribute is a foreign key, which suggests that multiple venue IDs can be stored in a single record, violating atomicity.
- Violation of 2NF: Assuming `venue_id` is functionally dependent on `movie_id`, there might be partial dependency if `venue_id` directly depends on any other non-key attribute.
- Violation of 3NF: Assuming `city` is functionally dependent on `venue_id`, and `venue_id` is functionally dependent on `movie_id`, there might be a transitive dependency.

6. Seats, Movie_Seats, and Movie_viewers Tables:
- No apparent violations. These tables primarily serve as junction tables to establish many-to-many relationships between movies, customers, and seats.

7. Concert and Concert_viewers Tables:
- No apparent violations. Both tables seem to adhere to 1NF, 2NF, and 3NF.

8. Standup_Comedy, Standup_seats, Standup_Seats_booked, and Standup_viewers Tables:
- Similar to the Movie tables, there might be violations depending on the dependencies between attributes.

9. Art_Exhibition and Exhibition_viewers Tables:
- Similar to the Concert tables, no apparent violations are observed.

10. SPORTS, Stadium_seats, Sport_Seats_booked, and Sports_viewers Tables:
- Similar to the Movie and Standup_Comedy tables, there might be violations depending on the dependencies between attributes.

Based on this analysis, further examination is required for the Movie, Standup_Comedy, and Sports-related tables to determine if normalization norms are violated.

DECOMPOSED TABLES:

A.)Admin Table:

Attribute	Functional Dependency	Type
admin_id	-> full_name	Primary Key
	-> username	Unique
	-> password_hash	
	-> email	Unique

B.)Customer_info (Customer_info):

Attribute	Functional Dependency	Type
cust_id	-> full_name	Primary Key
	-> username	Unique
	-> password_hash	
	-> dob	
	-> city	
	-> phone_no	
	-> email	Unique

C.)Event_list (Event_list):

Attribute	Functional Dependency	Type
event_id	-> event_type	Primary Key
	-> event_title	
	-> event_date	
	-> city	
	-> venue_id	Foreign Key

D.)Venue_list (Venue_list):

Attribute	Functional Dependency	Type
venue_id	-> venue_name	Primary Key
	-> city	
	-> capacity	

E.)Movie (Movie):

Attribute	Functional Dependency	Type
movie_id	-> movie_name	Primary Key
	-> date	
	-> venue_id	Foreign Key
	-> city	

F.)Seats (Seats):

Attribute	Functional Dependency	Type
seat_id	-> seat_name	Primary Key

G.)Movie_Seats (Movie_Seats):

Attribute	Functional Dependency	Type
movie_id	-> cust_id	Primary Key (Composite)
	-> seat_id	Foreign Key

H.)Movie_viewers (Movie_viewers):

Attribute	Functional Dependency	Type
movie_id	-> cust_id	Primary Key (Composite)
	-> customer_name	
	-> cinema_name	

I.)Concert (Concert):

Attribute	Functional Dependency	Type
concert_id	-> event_name	Primary Key
	-> date	
	-> venue_id	Foreign Key
	-> city	

J.)Concert_viewers (Concert_viewers):

Attribute	Functional Dependency	Type
concert_id	-> cust_id	Primary Key (Composite)
	-> customer_name	
	-> concert_name	

	-> tickets	

K.)Exhibition_viewers:

Attribute	Functional Dependency	Type
`exhibition_id`	-> cust_id	Primary Key (Composite)
	-> customer_name	
	-> `exhibition_name`	
	-> tickets	
	-> total_cost	

L.)Sports_viewers:

Attribute	Functional Dependency	Type
`sport_id`	-> cust_id	Primary Key (Composite)
	-> customer_name	
	-> sport_name	
	-> tickets	
	-> total_cost	

CREATION OF DATABASE

A.)DATABASE CREATION:

```
create database ticketHub;
use ticketHub;
```

B.)ADMIN TABLE

```
CREATE TABLE admin (
   admin_id INT PRIMARY KEY AUTO_INCREMENT,
   full_name VARCHAR(100) NOT NULL,
   username VARCHAR(50) UNIQUE NOT NULL,
   password_hash VARCHAR(255) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL
   );

desc admin;
```

Field	Type	Null	Key	Default	Extra
admin_id	int	NO	PRI	NULL	auto_increment
full_name	varchar(100)	NO		NULL	
username	varchar(50)	NO	UNI	NULL	
password_hash	varchar(255)	NO		NULL	
email	varchar(100)	NO	UNI	NULL	

C.)CUSTOMER_INFO.

TABLE

```
CREATE TABLE customer_info (
    cust_id INT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(100) NOT NULL,
    username VARCHAR(50) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    dob DATE,
    city VARCHAR(100),
    phone_no VARCHAR(15),
    email VARCHAR(100) UNIQUE NOT NULL
);
```

Field	Type	Null	Key	Default	Extra
cust_id	int	NO	PRI	NULL	auto_increment
full_name	varchar(100)	NO		NULL	
username	varchar(50)	NO	UNI	NULL	
password_hash	varchar(255)	NO		NULL	
dob	date	YES		NULL	
city	varchar(100)	YES		NULL	
phone_no	varchar(15)	YES		NULL	
email	varchar(100)	NO	UNI	NULL	

VALUES:

INSERT INTO customer_info (cust_id, full_name, username, password_hash, dob, city, phone_no, email) VALUES

(1, 'Aarav Patel', 'aarav_p', 'password123', '1995-03-10', 'Ahmedabad', '1234567890', 'aarav@example.com'),

(2, 'Neha Shah', 'neha_shr', 'securepwd', '1990-07-05', 'Surat', '9876543210', 'neha@gmail.com'),

(3, 'Rahul Desai', 'rahul_d', 'mysecretpass', '1988-11-22', 'Vadodara', '9876543211', 'rahul@example.com'),

(4, 'Priya Joshi', 'priya_j', 'p@ssw0rd', '1992-05-15', 'Rajkot', '9876543212', 'priya@example.com'),

(5, 'Manisha Patel', 'manisha_p', 'letmein', '1993-09-30', 'Surat', '9876543213', 'manisha@example.com'),

(6, 'Dinesh Mehta', 'dinesh_m', 'dinesh123', '1991-08-20', 'Ahmedabad', '9876543214', 'dinesh@example.com'),

(7, 'Anjali Shah', 'anjali_m_s', 'anjalipass', '1994-04-18', 'Vadodara', '9876543215', 'anjali@example.com'),

(8, 'Raj Patel', 'raj_p', 'rajp@ss', '1996-01-25', 'Surat', '9876543216', 'raj@example.com'),

(9, 'Komal Patel', 'komal_p', 'komal123', '1990-12-12', 'Ahmedabad', '9876543217', 'komal@example.com'),

(10, 'Suresh Dave', 'suresh_d', 'suresh123', '1993-06-28', 'Rajkot', '9876543218', 'suresh@example.com'),

(11, 'Shilpa Patel', 'shilpa_p', 'shilpa@123', '1989-10-15', 'Vadodara', '9876543219', 'shilpa@example.com'),

(12, 'Ravi Singh', 'ravi_s', 'ravi123', '1994-08-05', 'Delhi', '9876543220', 'ravi@example.com'),

(13, 'Aditi Sharma', 'aditi_s', 'aditi@123', '1991-02-20', 'Mumbai', '9876543221', 'aditi@example.com'),

(14, 'Ajay Kumar', 'ajay_k', 'ajay@123', '1990-11-12', 'Bangalore', '9876543222', 'ajay@example.com'),

(15, 'Sneha Gupta', 'sneha_gm', 'sneha@123', '1993-07-18', 'Chennai', '9876543223', 'sneha@example.com'),

(16, 'Vikram Singh', 'vikram_s', 'vikram@123', '1988-05-25', 'Hyderabad', '9876543224', 'vikram@example.com'),

(17, 'Deepika Nair', 'deepika_n', 'deepika@123', '1995-03-15', 'Kolkata', '9876543225', 'deepika@example.com'),

(18, 'Arjun Patel', 'arjun_p', 'arjun@123', '1992-09-30', 'Pune', '9876543226', 'arjun@example.com'),

(19, 'Meera Reddy', 'meera_r', 'meera@123', '1987-11-28', 'Jaipur', '9876543227', 'meera@example.com'),

(20, 'Rohit Gupta', 'rohit_g', 'rohit@123', '1996-04-22', 'Lucknow', '9876543228', 'rohit@example.com'),

(21, 'Akash Kumar', 'akash_k', 'akash@123', '1990-06-15', 'Patna', '9876543229', 'akash@example.com'),

(22, 'Riya Sharma', 'riya_sr', 'riya@123', '1993-09-25', 'Lucknow', '9876543230', 'riya@example.com'),

(23, 'Sanjay Patel', 'sanjay_p', 'sanjay@123', '1988-07-20', 'Varanasi', '9876543231', 'sanjay@example.com'),

(24, 'Jyoti Singh', 'jyoti_s', 'jyoti@123', '1995-05-12', 'Kanpur', '9876543232', 'jyoti@example.com'),

(25, 'Vishal Gupta', 'vishal_g', 'vishal@123', '1992-11-10', 'Allahabad', '9876543233', 'vishal@example.com'),

(26, 'Neha Tiwari', 'neha_t', 'neha@123', '1991-03-28', 'Agra', '9876543234', 'neha@example.com'),

(27, 'Sachin Singh', 'sachin_ss', 'sachin@123', '1989-08-22', 'Bhopal', '9876543235', 'sachin@example.com'),

(28, 'pooja_sa', 'pooja@123', '1994-04-15', 'Indore', '9876543236', 'pooja@example.com'),

(29, 'Alok Mishra', 'alok_m', 'alok@123', '1993-01-30', 'Jaipur', '9876543237', 'alok@example.com'),

(30, 'Kirti Gupta', 'kirti_g', 'kirti@123', '1990-10-12', 'Lucknow', '9876543238', 'kirti@example.com'),

(31, 'Rakesh Verma', 'rakesh_v', 'rakesh@123', '1996-07-18', 'Chandigarh', '9876543239', 'rakesh@example.com'),

(32, 'Sapna Singh', 'sapna_s', 'sapna@123', '1988-05-25', 'Nagpur', '9876543240', 'sapna@example.com'),

(33, 'Amit Kumar', 'amit_k', 'amit@123', '1992-03-15', 'Ahmedabad', '9876543241', 'amit@example.com'),

(34, 'Vikas Gupta', 'vikas_g', 'vikas@123', '1990-09-30', 'Surat', '9876543242', 'vikas@example.com'),

(35, 'Preeti Sharma', 'preeti_sm', 'preeti@123', '1987-11-28', 'Mumbai', '9876543243', 'preeti@example.com'),

(36, 'Rahul Yadav', 'rahul_y', 'rahul@123', '1996-04-22', 'Chennai', '9876543244', 'rahul@gmail.com'),

(37, 'Ananya Singh', 'ananya_sghr', 'ananya@123', '1992-09-30', 'Hyderabad', '9876543245', 'ananya@example.com'),

(38, 'Arun Patel', 'arun_p', 'arun@123', '1985-06-20', 'Pune', '9876543246', 'arun@example.com'),

(39, 'Monika Gupta', 'monika_g', 'monika@123', '1993-07-28', 'Delhi', '9876543247', 'monika@example.com'),

(40, 'Rohan Sharma', 'rohanh_s', 'rohan@123', '1991-02-22', 'Bangalore', '9876543248', 'rohan@example.com'),

(41, 'Divya Patel', 'divya_p', 'divya@123', '1994-09-15', 'Ahmedabad', '9876543249', 'divya@example.com'),

(42, 'Vivek Singh', 'vivek_s', 'vivek@123', '1989-12-25', 'Varanasi', '9876543250', 'vivek@example.com'),

(43, 'Riya Gupta', 'riya_g', 'riya@123', '1996-08-22', 'Delhi', '9876543251', 'riya@gmail.com'),

(44, 'Karan Sharma', 'karan_s', 'karan@123', '1993-04-18', 'Mumbai', '9876543252', 'karan@example.com'),

(45, 'Ananya Mishra', 'ananya_m', 'ananya@123', '1990-11-30', 'Kolkata', '9876543253', 'ananya123@gmail.com'),

(46, 'Rahul Kapoor', 'rahul_k', 'rahul@123', '1987-07-15', 'Chennai', '9876543254', 'rahul_kapoor23@yahoo.com'),

(47, 'Sneha Reddy', 'sneha_r', 'sneha@123', '1995-02-20', 'Hyderabad', '9876543255', 'sneha_red_s@gmail.com'),

(48, 'Ankit Kumar', 'ankit_k', 'ankit@123', '1992-10-12', 'Pune', '9876543256', 'ankit123@outlook.com'),

(49, 'Kriti Sharma', 'kriti_s', 'kriti@123', '1988-06-28', 'Lucknow', '9876543257', 'kriti_sharma_88@yahoo.com'),

(50, 'Deepak Verma', 'deepak_v', 'deepak@123', '1991-03-22', 'Bangalore', '9876543258', 'verma_deepak007@hotmail.com'),

(51, 'Nidhi Gupta', 'nidhi_g', 'nidhi@123', '1989-09-18', 'Delhi', '9876543259', 'nidhi_gupta1989@rediffmail.com'),

(52, 'Rohan Singh', 'rohan_s', 'rohan@123', '1993-06-10', 'Mumbai', '9876543260', 'rohan_777@live.com'),

(53, 'Komal Sharma', 'komal_snt', 'komal@123', '1995-01-05', 'Ahmedabad', '9876543261', 'komal_sharma_95@gmail.com'),

(54, 'Amit Singh', 'amit_sg', 'amit@123', '1994-08-28', 'Surat', '9876543262', 'singh_amit_94@yahoo.com'),

(55, 'Shivani Patel', 'shivani_p', 'shivani@123', '1988-04-15', 'Jaipur', '9876543263', 'shivani123@live.com'),

(56, 'Akash Gupta', 'akash_g', 'akash@123', '1992-12-20', 'Chandigarh', '9876543264', 'akash_gupta_1992@gmail.com'),

(57, 'Prachi Sharma', 'prachi_s', 'prachi@123', '1996-05-22', 'Nagpur', '9876543265', 'prachi_22@yahoo.com'),

(58, 'Sumit Yadav', 'sumit_y', 'sumit@123', '1990-10-30', 'Kolkata', '9876543266', 'sumit_yadav_90@hotmail.com'),

(59, 'Anjali Verma', 'anjali_v', 'anjali@123', '1987-02-18', 'Hyderabad', '9876543267', 'anjali.verma_87@gmail.com'),

(60, 'Siddharth Gupta', 'siddharth_g', 'siddharth@123', '1993-09-05', 'Pune', '9876543268', 'gupta_sidd_93@yahoo.com'),

(61, 'Mansi Sharma', 'mansi_s', 'mansi@123', '1991-11-12', 'Delhi', '9876543269', 'mansi_sharma91@outlook.com'),

(62, 'Rohit Verma', 'rohit_v', 'rohit@123', '1989-07-28', 'Mumbai', '9876543270', 'rohit_verma89@rediffmail.com'),

(63, 'Anjali Singh', 'anjali_singh', 'anjali@123', '1994-03-22', 'Lucknow', '9876543271', 'singh.anjali94@hotmail.com'),

(64, 'Sneha Gupta', 'sneha_ggg', 'sneha@123', '1997-08-15', 'Chennai', '9876543272', 'sneha_1997@gmail.com'),

(65, 'Alok Tiwari', 'alok_t', 'alok@123', '1992-06-20', 'Bangalore', '9876543273', 'tiwari_alok92@live.com'),

(66, 'Divya Singh', 'divya_s', 'divya@123', '1995-01-30', 'Ahmedabad', '9876543274', 'singh_divya95@gmail.com'),

(67, 'Rakesh Sharma', 'rakesh_s', 'rakesh@123', '1990-09-18', 'Surat', '9876543275', 'rakesh_90@yahoo.com'),

(68, 'Kirti Gupta', 'kirti_gm', 'kirti@123', '1988-03-25', 'Vadodara', '9876543276', 'kirti.gupta88@outlook.com'),

(69, 'Arjun Kumar', 'arjun_k', 'arjun@123', '1993-07-12', 'Rajkot', '9876543277', 'arjun.kumar93@gmail.com'),

(70, 'Monika Singh', 'monika_s', 'monika@123', '1991-10-28', 'Pune', '9876543278', 'singh.monika91@live.com'),

(71, 'Rohit Kumar', 'rohit_k', 'rohit@123', '1989-05-22', 'Jaipur', '9876543279', 'rohit.kumar89@yahoo.com'),

(72, 'Sneha Verma', 'sneha_v', 'sneha@123', '1994-04-15', 'Chandigarh', '9876543280', 'sneha_verma94@outlook.com'),

(73, 'Ankit Gupta', 'ankit_g', 'ankit@123', '1996-08-30', 'Nagpur', '9876543281', 'ankit.gupta96@gmail.com'),

(74, 'Pooja Sharma', 'pooja_s', 'pooja@123', '1992-12-18', 'Kolkata', '9876543282', 'pooja.sharma92@yahoo.com'),

(75, 'Vikas Singh', 'vikas_s', 'vikas@123', '1988-02-10', 'Hyderabad', '9876543283', 'singh.vikas88@live.com'),

(76, 'Neha Yadav', 'neha_y', 'neha@123', '1995-06-22', 'Delhi', '9876543284', 'neha.yadav95@gmail.com'),

(77, 'Alok Gupta', 'alok_g', 'alok@123', '1990-11-30', 'Mumbai', '9876543285', 'alok.gupta90@yahoo.com'),

(78, 'Preeti Singh', 'preeti_s', 'preeti@123', '1987-08-15', 'Lucknow', '9876543286', 'preeti.singh87@outlook.com'),

(79, 'Anjali Sharma', 'anjali_shy', 'anjali@123', '1993-03-20', 'Chennai', '9876543287', 'sharma.anjali93@gmail.com'),

(80, 'Rahul Verma', 'rahul_v', 'rahul@123', '1989-09-12', 'Bangalore', '9876543288', 'verma.rahul89@yahoo.com'),

(81, 'Komal Singh', 'komal_s', 'komal@123', '1996-07-28', 'Ahmedabad', '9876543289', 'singh.komal96@live.com'),

(82, 'Ravi Kumar', 'ravi_k', 'ravi@123', '1991-04-05', 'Surat', '9876543290', 'ravi.kumar91@gmail.com'),

(83, 'Ananya Gupta', 'ananya_g', 'ananya@123', '1990-11-18', 'Vadodara', '9876543291', 'ananya.gupta90@yahoo.com'),

(84, 'Sachin Sharma', 'sachin_s', 'sachin@123', '1994-02-22', 'Rajkot', '9876543292', 'sharma.sachin94@outlook.com'),

(85, 'Riya Singh', 'riya_s', 'riya@123', '1988-07-30', 'Pune', '9876543293', 'riya.singh88@gmail.com'),

(86, 'Amit Sharma', 'amit_s', 'amit@123', '1993-10-15', 'Jaipur', '9876543294', 'sharma.amit93@yahoo.com'),

(87, 'Deepika Verma', 'deepika_v', 'deepika@123', '1995-03-18', 'Chandigarh', '9876543295', 'verma.deepika95@gmail.com'),

(88, 'Ajay Singh', 'ajay_s', 'ajay@123', '1990-09-25', 'Nagpur', '9876543296', 'ajay.singh90@yahoo.com'),

(89, 'Neha Sharma', 'neha_s', 'neha@123', '1988-11-30', 'Kolkata', '9876543297', 'sharma.neha88@outlook.com'),

(90, 'Arun Kumar', 'arun_k', 'arun@123', '1993-05-12', 'Hyderabad', '9876543298', 'arun.kumar93@gmail.com'),

(91, 'Kriti Verma', 'kriti_v', 'kriti@123', '1989-10-28', 'Delhi', '9876543299', 'verma.kriti89@yahoo.com'),

(92, 'Rahul Sharma', 'rahul_s', 'rahul@123', '1996-04-05', 'Mumbai', '9876543300', 'sharma.rahul96@outlook.com'),

(93, 'Anjali Singh', 'anjali_s', 'anjali@123', '1991-11-15', 'Lucknow', '9876543301', 'singh.anjali91@gmail.com'),

(94, 'Vikas Verma', 'vikas_v', 'vikas@123', '1987-08-22', 'Chennai', '9876543302', 'verma.vikas87@yahoo.com'),

(95, 'Sneha Sharma', 'sneha_s', 'sneha@123', '1994-02-28', 'Bangalore', '9876543303', 'sharma.sneha94@outlook.com'),

(96, 'Rohit Singh', 'rohit_s', 'rohit@123', '1992-07-12', 'Ahmedabad', '9876543304', 'singh.rohit92@gmail.com'),

(97, 'Pooja Verma', 'pooja_v', 'pooja@123', '1988-12-25', 'Surat', '9876543305', 'verma.pooja88@yahoo.com'),

(98, 'Alok Singh', 'alok_s', 'alok@123', '1995-09-30', 'Vadodara', '9876543306', 'singh.alok95@outlook.com'),

(99, 'Ananya Sharma', 'ananya_s', 'ananya@123', '1990-04-15', 'Rajkot', '9876543307', 'sharma.ananya90@gmail.com'),

(100, 'Ravi Verma', 'ravi_v', 'ravi@123', '1986-11-22', 'Pune', '9876543308', 'verma.ravi86@yahoo.com'),

(101, 'Komal Singh', 'komal_sp', 'komal@123', '1993-08-30', 'Jaipur', '9876543309', 'singh.komal93@gmail.com'),

(102, 'Amit Verma', 'amit_v', 'amit@123', '1989-12-15', 'Chandigarh', '9876543310', 'verma.amit89@yahoo.com'),

(103, 'Neha Singh', 'neha_sy', 'neha@123', '1996-06-22', 'Nagpur', '9876543311', 'singh.neha96@outlook.com'),

(104, 'Arjun Sharma', 'arjun_s', 'arjun@123', '1991-01-28', 'Kolkata', '9876543312', 'sharma.arjun91@gmail.com'),

(105, 'Anjali Verma', 'anjali_vt', 'anjali@123', '1988-07-10', 'Hyderabad', '9876543313', 'verma.anjali88@yahoo.com'),

(106, 'Rahul Singh', 'rahul_sj', 'rahul@123', '1994-04-18', 'Delhi', '9876543314', 'singh.rahul94@gmail.com');

cust_id	full_name	username	password_hash	dob	city	phone_no	email
1	Aarav Patel	aarav_p	password123	1995-03-10	Ahmedabad	1234567890	aarav@example.com
2	Neha Shah	neha_shr	securepwd	1990-07-05	Surat	9876543210	neha@gmail.com
3	Rahul Desai	rahul_d	mysecretpass	1988-11-22	Vadodara	9876543211	rahul@example.com
4	Priya Joshi	priya_j	p@ssw0rd	1992-05-15	Rajkot	9876543212	priya@example.com
5	Manisha Patel	manisha_p	letmein	1993-09-30	Surat	9876543213	manisha@example.com
6	Dinesh Mehta	dinesh_m	dinesh123	1991-08-20	Ahmedabad	9876543214	dinesh@example.com
7	Anjali Shah	anjali_m_s	anjalipass	1994-04-18	Vadodara	9876543215	anjali@example.com
8	Raj Patel	raj_p	rajp@ss	1996-01-25	Surat	9876543216	raj@example.com
9	Komal Patel	komal_p	komal123	1990-12-12	Ahmedabad	9876543217	komal@example.com
10	Suresh Dave	suresh_d	suresh123	1993-06-28	Rajkot	9876543218	suresh@example.com
11	Shilpa Patel	shilpa_p	shilpa@123	1989-10-15	Vadodara	9876543219	shilpa@example.com
12	Ravi Singh	ravi_s	ravi123	1994-08-05	Delhi	9876543220	ravi@example.com
13	Aditi Sharma	aditi_s	aditi@123	1991-02-20	Mumbai	9876543221	aditi@example.com
14	Ajay Kumar	ajay_k	ajay@123	1990-11-12	Bangalore	9876543222	ajay@example.com
15	Sneha Gupta	sneha_gm	sneha@123	1993-07-18	Chennai	9876543223	sneha@example.com
16	Vikram Singh	vikram_s	vikram@123	1988-05-25	Hyderabad	9876543224	vikram@example.com
17	Deepika Nair	deepika_n	deepika@123	1995-03-15	Kolkata	9876543225	deepika@example.com

D.)EVENT-LIST

TABLE

```
CREATE TABLE event_list (
    event_id INT PRIMARY KEY AUTO_INCREMENT,
    event_type ENUM('concert', 'movie', 'art-exhibition', 'standupcomedy', 'sports'),
    event_title VARCHAR(255) NOT NULL,
    event_date DATE,
    city VARCHAR(100),
    venue_id INT,
    ticket_price DECIMAL(10, 2)
```

FOREIGN KEY (venue_id) REFERENCES venue_list(id));

Field	Type	Null	Key	Default	Extra
event_id	int	NO	PRI	NULL	auto_increment
event_type	enum('concert','movie','art-exhibition','standupc...	YES		NULL	
event_title	varchar(255)	NO		NULL	
event_date	date	YES		NULL	
city	varchar(100)	YES		NULL	
venue_id	int	YES	UNI	NULL	
ticket_price	decimal(10,2)	YES		NULL	

VALUES

INSERT INTO event_list (event_type, event_title, event_date, city, venue_id, ticket_price) VALUES

('concert', 'Rock On Festival', '2024-04-15', 'Mumbai', 1, 500.00),

('movie', 'The Dark Knight Screening', '2024-04-20', 'Delhi', 2, 300.00),

('art-exhibition', 'Modern Art Showcase', '2024-05-05', 'Kolkata', 3, 100.00),

('standupcomedy', 'Laugh Riot Live', '2024-05-10', 'Bangalore', 4, 250.00),

('sports', 'Cricket Match: India vs. Australia', '2024-05-20', 'Chennai', 5, 400.00),

('concert', 'Bollywood Beats Night', '2024-06-05', 'Pune', 6, 450.00),

('movie', 'Avengers: Endgame Screening', '2024-06-15', 'Hyderabad', 7, 350.00),

('art-exhibition', 'Sculpture Exhibition', '2024-07-10', 'Ahmedabad', 8, 150.00),

('standupcomedy', 'Comedy Central Live', '2024-07-20', 'Surat', 9, 200.00),

('sports', 'Football Match: Manchester United vs. Liverpool', '2024-08-05', 'Lucknow', 10, 300.00),

('concert', 'EDM Night: Tomorrowland Experience', '2024-08-15', 'Jaipur', 11, 600.00),

('movie', 'Inception Screening', '2024-09-10', 'Varanasi', 12, 250.00),

('art-exhibition', 'Renaissance Art Show', '2024-09-20', 'Bhopal', 13, 120.00),

('standupcomedy', 'Hilarious Hilarity Tour', '2024-10-05', 'Indore', 14, 180.00),

('sports', 'Basketball Game: NBA All-Stars', '2024-10-15', 'Guwahati', 15, 350.00),

('concert', 'Pop Sensation Concert', '2024-11-10', 'Chandigarh', 16, 500.00),

('movie', 'Interstellar Screening', '2024-11-20', 'Nagpur', 17, 300.00),

('art-exhibition', 'Graffiti Art Showcase', '2024-12-05', 'Kochi', 18, 100.00),

('standupcomedy', 'Laugh Fest: The Comedy Show', '2024-12-15', 'Bhubaneswar', 19, 220.00),

('sports', 'Tennis Tournament: Wimbledon Finals', '2025-01-10', 'Agra', 20, 400.00),

('concert', 'Classical Music Night', '2025-01-20', 'Goa', 21, 550.00),

('movie', 'The Godfather Screening', '2025-02-05', 'Udaipur', 22, 280.00),

('art-exhibition', 'Photography Exhibition', '2025-02-15', 'Shimla', 23, 130.00),

('standupcomedy', 'Comedy Carnival', '2025-03-10', 'Dehradun', 24, 190.00),

('sports', 'Golf Championship', '2025-03-20', 'Mysore', 25, 380.00),

('concert', 'Jazz Night', '2025-04-05', 'Vadodara', 26, 500.00),

('movie', 'Pulp Fiction Screening', '2025-04-15', 'Kanpur', 27, 320.00),

('art-exhibition', 'Abstract Art Expo', '2025-05-10', 'Ranchi', 28, 110.00),

('standupcomedy', 'Stand-Up Smackdown', '2025-05-20', 'Jodhpur', 29, 230.00),

('sports', 'Hockey Match: India vs. Pakistan', '2025-06-05', 'Visakhapatnam', 30, 450.00),

('concert', 'Indie Music Fest', '2025-06-15', 'Thiruvananthapuram', 31, 600.00),

('movie', 'The Shawshank Redemption Screening', '2025-07-10', 'Bikaner', 32, 300.00),

('art-exhibition', 'Cultural Art Fair', '2025-07-20', 'Jalandhar', 33, 140.00),

('standupcomedy', 'Giggle Gala', '2025-08-05', 'Raipur', 34, 200.00),

('sports', 'Formula 1 Grand Prix', '2025-08-15', 'Amritsar', 35, 500.00),

('concert', 'Rocktoberfest', '2025-09-10', 'Patna', 36, 450.00),

('movie', 'The Matrix Screening', '2025-09-20', 'Srinagar', 37, 280.00),

('art-exhibition', 'Surrealism Showcase', '2025-10-05', 'Guwahati', 38, 120.00),

('standupcomedy', 'Laugh-A-Thon', '2025-10-15', 'Shillong', 39, 180.00),

('sports', 'Rugby Match: World Cup Finals', '2025-11-10', 'Gangtok', 40, 350.00),

('concert', 'Country Music Night', '2025-11-20', 'Kohima', 41, 500.00),

('movie', 'Fight Club Screening', '2025-12-05', 'Agartala', 42, 320.00),

('art-exhibition', 'Street Art Festival', '2025-12-15', 'Itanagar', 43, 100.00),

('standupcomedy', 'Funny Fiesta', '2026-01-10', 'Imphal', 44, 220.00),

('sports', 'Boxing Championship', '2026-01-20', 'Aizawl', 45, 400.00),

('concert', 'Reggae Night', '2026-02-05', 'Kavaratti', 46, 550.00),

('movie', 'Forrest Gump Screening', '2026-02-15', 'Daman', 47, 280.00),

('art-exhibition', 'Impressionism Expo', '2026-03-10', 'Karaikal', 48, 130.00),

('standupcomedy', 'Joke Jam', '2026-03-20', 'Pondicherry', 49, 190.00),

('sports', 'Badminton Tournament: All India Open', '2026-04-05', 'Panaji', 50, 380.00);

event_id	event_type	event_title	event_date	city	venue_id	ticket_price
1	concert	Rock On Festival	2024-04-15	Mumbai	1	500.00
2	movie	The Dark Knight Screening	2024-04-20	Delhi	2	300.00
3	art-exhibition	Modern Art Showcase	2024-05-05	Kolkata	3	100.00
4	standupcomedy	Laugh Riot Live	2024-05-10	Bangalore	4	250.00
5	sports	Cricket Match: India vs. Australia	2024-05-20	Chennai	5	400.00
6	concert	Bollywood Beats Night	2024-06-05	Pune	6	450.00
7	movie	Avengers: Endgame Screening	2024-06-15	Hyderabad	7	350.00
8	art-exhibition	Sculpture Exhibition	2024-07-10	Ahmedabad	8	150.00
9	standupcomedy	Comedy Central Live	2024-07-20	Surat	9	200.00
10	sports	Football Match: Manchester United vs. Liverpool	2024-08-05	Lucknow	10	300.00
11	concert	EDM Night: Tomorrowland Experience	2024-08-15	Jaipur	11	600.00

D.)VENUE_LIST

TABLE

NOTE:AMENITIES MULTIVALUED

```
CREATE TABLE venue_list (
    venue_id INT PRIMARY KEY AUTO_INCREMENT,
    venue_name VARCHAR(100) NOT NULL,
    city VARCHAR(100),
    capacity INT,
    amenities VARCHAR(255),
```

```
    FOREIGN KEY (venue_id) REFERENCES event_list(venue_id)
);
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
venue_name	varchar(100)	NO		NULL	
city	varchar(100)	YES		NULL	
capacity	int	YES		NULL	
amenities	varchar(255)	YES		NULL	

VALUES

```
INSERT INTO venue_list (venue_name, city, capacity, amenities) VALUES
('Mumbai Arena', 'Mumbai', 10000, 'Parking, Food Court, Restrooms'),
('Delhi Convention Center', 'Delhi', 8000, 'Parking, Cafeteria, WiFi'),
('Kolkata Art Gallery', 'Kolkata', 5000, 'Parking, Gallery Space, Restrooms'),
('Bangalore Comedy Club', 'Bangalore', 3000, 'Parking, Bar, Seating Area'),
('Chennai Stadium', 'Chennai', 15000, 'Parking, Concession Stands, Restrooms'),
('Pune Amphitheater', 'Pune', 6000, 'Parking, Stage, Seating'),
('Hyderabad Cinema Complex', 'Hyderabad', 7000, 'Parking, Multiplex Screens, Snack Bar'),
('Ahmedabad Exhibition Hall', 'Ahmedabad', 4000, 'Parking, Exhibition Space, Restrooms'),
('Surat Comedy Lounge', 'Surat', 2500, 'Parking, Lounge Area, Stage'),
('Lucknow Sports Arena', 'Lucknow', 12000, 'Parking, Seating, Locker Rooms'),
('Jaipur Music Hall', 'Jaipur', 5500, 'Parking, Stage, Acoustics'),
('Varanasi Cinema Plaza', 'Varanasi', 8000, 'Parking, Multiplex Screens, Snack Bar'),
('Bhopal Cultural Center', 'Bhopal', 3500, 'Parking, Exhibition Space, Restrooms'),
('Indore Comedy Zone', 'Indore', 2000, 'Parking, Bar, Seating Area'),
('Guwahati Stadium', 'Guwahati', 13000, 'Parking, Concession Stands, Restrooms'),
('Chandigarh Music Dome', 'Chandigarh', 6000, 'Parking, Stage, Seating'),
('Nagpur Cinema Square', 'Nagpur', 7500, 'Parking, Multiplex Screens, Snack Bar'),
```

('Kochi Art Pavilion', 'Kochi', 4500, 'Parking, Gallery Space, Restrooms'),

('Jodhpur Comedy Castle', 'Jodhpur', 2800, 'Parking, Lounge Area, Stage'),

('Visakhapatnam Sports Complex', 'Visakhapatnam', 11000, 'Parking, Seating, Locker Rooms'),

('Thiruvananthapuram Jazz Hall', 'Thiruvananthapuram', 5000, 'Parking, Stage, Acoustics'),

('Bikaner Cinema Lounge', 'Bikaner', 7800, 'Parking, Multiplex Screens, Snack Bar'),

('Jalandhar Art Center', 'Jalandhar', 4000, 'Parking, Exhibition Space, Restrooms'),

('Raipur Comedy Hub', 'Raipur', 2200, 'Parking, Bar, Seating Area'),

('Amritsar Sports Stadium', 'Amritsar', 14000, 'Parking, Concession Stands, Restrooms'),

('Patna Music Hall', 'Patna', 6500, 'Parking, Stage, Seating'),

('Srinagar Cinema Plaza', 'Srinagar', 8200, 'Parking, Multiplex Screens, Snack Bar'),

('Guwahati Cultural Center', 'Guwahati', 3800, 'Parking, Exhibition Space, Restrooms'),

('Shillong Comedy Club', 'Shillong', 2300, 'Parking, Lounge Area, Stage'),

('Gangtok Stadium', 'Gangtok', 10000, 'Parking, Seating, Locker Rooms'),

('Kohima Music Dome', 'Kohima', 5700, 'Parking, Stage, Acoustics'),

('Agartala Cinema Square', 'Agartala', 7600, 'Parking, Multiplex Screens, Snack Bar'),

('Itanagar Art Pavilion', 'Itanagar', 4300, 'Parking, Gallery Space, Restrooms'),

('Imphal Comedy Castle', 'Imphal', 2500, 'Parking, Lounge Area, Stage'),

('Aizawl Sports Complex', 'Aizawl', 12000, 'Parking, Seating, Locker Rooms'),

('Kavaratti Music Hall', 'Kavaratti', 6200, 'Parking, Stage, Acoustics'),

('Daman Cinema Lounge', 'Daman', 8000, 'Parking, Multiplex Screens, Snack Bar'),

('Karaikal Art Center', 'Karaikal', 4700, 'Parking, Exhibition Space, Restrooms'),

('Pondicherry Comedy Hub', 'Pondicherry', 2400, 'Parking, Bar, Seating Area'),

('Panaji Sports Stadium', 'Panaji', 13000, 'Parking, Concession Stands, Restrooms');

id	venue_name	city	capacity	amenities
1	Mumbai Arena	Mumbai	10000	Parking, Food Court, Restrooms
2	Delhi Convention Center	Delhi	8000	Parking, Cafeteria, WiFi
3	Kolkata Art Gallery	Kolkata	5000	Parking, Gallery Space, Restrooms
4	Bangalore Comedy Club	Bangalore	3000	Parking, Bar, Seating Area
5	Chennai Stadium	Chennai	15000	Parking, Concession Stands, Restrooms
6	Pune Amphitheater	Pune	6000	Parking, Stage, Seating
7	Hyderabad Cinema Complex	Hyderabad	7000	Parking, Multiplex Screens, Snack Bar
8	Ahmedabad Exhibition Hall	Ahmedabad	4000	Parking, Exhibition Space, Restrooms
9	Surat Comedy Lounge	Surat	2500	Parking, Lounge Area, Stage
10	Lucknow Sports Arena	Lucknow	12000	Parking, Seating, Locker Rooms
11	Jaipur Music Hall	Jaipur	5500	Parking, Stage, Acoustics

D.)MOVIE

TABLE

```
CREATE TABLE Movie (
    movie_id VARCHAR(255) PRIMARY KEY,
    movie_name VARCHAR(255),
    date DATE,
    venue_id int,
    city VARCHAR(255),
    -- no_of_seats INT,
--     seat_name VARCHAR(255),
    cost_per_ticket DECIMAL(10, 2),
    -- total_cost DECIMAL(10, 2),
    foreign key (venue_id) references event_list(event_id)

);
```

Field	Type	Null	Key	Default	Extra
movie_id	varchar(255)	NO	PRI	NULL	
movie_name	varchar(255)	YES		NULL	
date	date	YES		NULL	
venue_id	int	YES	MUL	NULL	
city	varchar(255)	YES		NULL	
cost_per_ticket	decimal(10,2)	YES		NULL	

VALUES

```
INSERT INTO movie (movie_id, movie_name, DATE, city, venue_id, cost_per_ticket)
SELECT
    event_id,
    event_title,
    event_date,
    city,
    venue_id,
    ticket_price
FROM
    event_list
WHERE
    event_type = 'movie';


select * from movie;
```

movie_id	movie_name	date	venue_id	city	cost_per_ticket
12	Inception Screening	2024-09-10	12	Varanasi	250.00
17	Interstellar Screening	2024-11-20	17	Nagpur	300.00
2	The Dark Knight Screening	2024-04-20	2	Delhi	300.00
22	The Godfather Screening	2025-02-05	22	Udaipur	280.00
27	Pulp Fiction Screening	2025-04-15	27	Kanpur	320.00
32	The Shawshank Redemption Screening	2025-07-10	32	Bikaner	300.00
37	The Matrix Screening	2025-09-20	37	Srinagar	280.00
42	Fight Club Screening	2025-12-05	42	Agartala	320.00
47	Forrest Gump Screening	2026-02-15	47	Daman	280.00
7	Avengers: Endgame Screening	2024-06-15	7	Hyderabad	350.00
NULL	NULL	NULL	NULL	NULL	NULL

```
CREATE TABLE Movie_viewers (
    movie_id INT,
    cust_id INT,
    customer_name VARCHAR(100),
    cinema_name VARCHAR(100),
    total_cost DECIMAL(10, 2),
    PRIMARY KEY (movie_id, cust_id)
);
```

Field	Type	Null	Key	Default	Extra
movie_id	int	NO	PRI	NULL	
cust_id	int	NO	PRI	NULL	
customer_name	varchar(100)	YES		NULL	
cinema_name	varchar(100)	YES		NULL	
total_cost	decimal(10,2)	YES		NULL	

UNDER MOVIE 3 MORE TABLES:

a.)Seats

b.)Movie_Seats

c.)Movie_viewers

a.)Seats:

TABLE

```
CREATE TABLE Seats (
    seat_id INT AUTO_INCREMENT PRIMARY KEY,
    seat_name VARCHAR(10)
);
```

Field	Type	Null	Key	Default	Extra
seat_id	int	NO	PRI	NULL	auto_increment
seat_name	varchar(10)	YES		NULL	

VALUES

```
INSERT INTO Seats (seat_name)
SELECT CONCAT(movie_letter, seat_number)
FROM
    (SELECT 'A' AS movie_letter UNION ALL
     SELECT 'B' UNION ALL
     SELECT 'C' UNION ALL
     SELECT 'D' UNION ALL
     SELECT 'E' UNION ALL
     SELECT 'F' UNION ALL
     SELECT 'G' UNION ALL
     SELECT 'H' UNION ALL
```

```
    SELECT 'I' UNION ALL
    SELECT 'J' UNION ALL
    SELECT 'K') AS movies
CROSS JOIN
   (SELECT 1 AS seat_number UNION ALL
    SELECT 2 UNION ALL
    SELECT 3 UNION ALL
    SELECT 4 UNION ALL
    SELECT 5 UNION ALL
    SELECT 6 UNION ALL
    SELECT 7 UNION ALL
    SELECT 8 UNION ALL
    SELECT 9 UNION ALL
    SELECT 10 UNION ALL
    SELECT 11 UNION ALL
    SELECT 12 UNION ALL
    SELECT 13 UNION ALL
    SELECT 14 UNION ALL
    SELECT 15 UNION ALL
    SELECT 16 UNION ALL
    SELECT 17 UNION ALL
    SELECT 18 UNION ALL
    SELECT 19 UNION ALL
    SELECT 20 UNION ALL
    SELECT 21 UNION ALL
    SELECT 22 UNION ALL
    SELECT 23 UNION ALL
```

```
    SELECT 24 UNION ALL
    SELECT 25 UNION ALL
    SELECT 26 UNION ALL
    SELECT 27 UNION ALL
    SELECT 28 UNION ALL
    SELECT 29 UNION ALL
    SELECT 30) AS seats;
```

seat_id	seat_name
1	K1
2	J1
3	I1
4	H1
5	G1
6	F1
7	E1
8	D1
9	C1
10	B1
11	A1

b.)Movie_Seats

TABLE

```
CREATE TABLE Movie_Seats (
    movie_id INT,
    cust_id INT,
    seat_id INT,
    PRIMARY KEY (movie_id, cust_id, seat_id),
    FOREIGN KEY (movie_id) REFERENCES Movie_viewers(movie_id),
    FOREIGN KEY (cust_id) REFERENCES Movie_viewers(cust_id),
    FOREIGN KEY (seat_id) REFERENCES Seats(seat_id)
);
```

Field	Type	Null	Key	Default	Extra
movie_id	int	YES		NULL	
cust_id	int	YES		NULL	
seat_id	int	YES	MUL	NULL	

VALUES

INSERT INTO Movie_Seats (movie_id, cust_id, seat_id) VALUES

(12, 23, 1), (12, 23, 2), (12, 23, 3), (12, 23, 4), (12, 23, 5),

(17, 88, 6), (17, 88, 7),

(2, 91, 8), (2, 91, 9), (2, 91, 10), (2, 91, 11),

(22, 98, 2), (22, 98, 5), (22, 98, 3),

(27, 75, 12),

(32, 6, 1), (32, 6, 2), (32, 6, 3), (32, 6, 4), (32, 6, 5),

(32, 6, 23), (32, 6, 24), (32, 6, 25), (32, 6, 26), (32, 6, 27),

(37, 81, 12), (37, 81, 13),

(42, 67, 14),

(47, 11, 1), (47, 11, 2), (47, 11, 3), (47, 11, 4), (47, 11, 5),

(47, 11, 16), (47, 11, 17), (47, 11, 18), (47, 11, 19), (47, 11, 20),

(73, 47, 21), (73, 47, 22);

movie_id	cust_id	seat_id
12	23	1
12	23	2
12	23	3
12	23	4
12	23	5
17	88	6
17	88	7
2	91	8
2	91	9
2	91	10

c.)Movie_viewers

TABLE

```
CREATE TABLE Movie_viewers (
    movie_id INT,
    cust_id INT,
    customer_name VARCHAR(100),
    cinema_name VARCHAR(100),
    total_cost DECIMAL(10, 2),
    PRIMARY KEY (movie_id, cust_id)
);
```

Field	Type	Null	Key	Default	Extra
movie_id	int	NO	PRI	NULL	
cust_id	int	NO	PRI	NULL	
customer_name	varchar(100)	YES		NULL	
cinema_name	varchar(100)	YES		NULL	
total_cost	decimal(10,2)	YES		NULL	

VALUES

INSERT INTO Movie_viewers (movie_id, cust_id, customer_name, cinema_name, total_cost) VALUES

(12, 23, 'Sanjay Patel', 'PVR', 1250),

(17, 88, 'Ajay Singh', 'INOX', 600),

(2, 91, 'Kriti Verma', 'PVR', 1200),

(22, 98, 'Alok Singh', 'PVR', 740),

(27, 75, 'Vikas Singh', 'INOX', 320),

(32, 6, 'Dinesh Mehta', 'INOX', 3000),

(37, 81, 'Komal Singh', 'INOX', 560),

(42, 67, 'Rakesh Sharma', 'Rajhans', 320),

(47, 11, 'Shilpa Patel', 'Rajhans', 2800),

(73, 47, 'Sneha Reddy', 'PVR', 700);

movie_id	cust_id	customer_name	cinema_name	total_cost
2	91	Kriti Verma	PVR	1200.00
12	23	Sanjay Patel	PVR	1250.00
17	88	Ajay Singh	INOX	600.00
22	98	Alok Singh	PVR	740.00
27	75	Vikas Singh	INOX	320.00
32	6	Dinesh Mehta	INOX	3000.00
37	81	Komal Singh	INOX	560.00
42	67	Rakesh Sharma	Rajhans	320.00
47	11	Shilpa Patel	Rajhans	2800.00
73	47	Sneha Reddy	PVR	700.00
NULL	NULL	NULL	NULL	NULL

E.)CONCERT

TABLE

CREATE TABLE CONCERT (

concert_id VARCHAR(255) PRIMARY KEY,

event_name VARCHAR(255),

```
    date DATE,
    venue_id int,
    city VARCHAR(255),
    cost_per_ticket DECIMAL(10, 2),
    foreign key (venue_id) references event_list(event_id)

);
```

Field	Type	Null	Key	Default	Extra
concert_id	varchar(255)	NO	PRI	NULL	
event_name	varchar(255)	YES		NULL	
date	date	YES		NULL	
venue_id	int	YES	MUL	NULL	
city	varchar(255)	YES		NULL	
cost_per_ticket	decimal(10,2)	YES		NULL	

VALUES

```
INSERT INTO concert (concert_id, event_name, DATE, city, venue_id, cost_per_ticket)
SELECT
    event_id,
    event_title,
    event_date,
    city,
    venue_id,
    ticket_price
FROM
    event_list
WHERE
    event_type = 'concert';
```

concert_id	event_name	date	venue_id	city	cost_per_ticket
1	Rock On Festival	2024-04-15	1	Mumbai	500.00
11	EDM Night: Tomorrowland Experience	2024-08-15	11	Jaipur	600.00
16	Pop Sensation Concert	2024-11-10	16	Chandigarh	500.00
21	Classical Music Night	2025-01-20	21	Goa	550.00
26	Jazz Night	2025-04-05	26	Vadodara	500.00
31	Indie Music Fest	2025-06-15	31	Thiruvananthapuram	600.00
36	Rocktoberfest	2025-09-10	36	Patna	450.00
41	Country Music Night	2025-11-20	41	Kohima	500.00
46	Reggae Night	2026-02-05	46	Kavaratti	550.00
6	Bollywood Beats Night	2024-06-05	6	Pune	450.00
NULL	NULL	NULL	NULL	NULL	NULL

b.)Concert_viewers

TABLE

```
CREATE TABLE concert_viewers (
    concert_id INT,
    cust_id INT,
    customer_name VARCHAR(100),
    concert_name VARCHAR(100),
    tickets int,
    total_cost DECIMAL(10, 2),
    PRIMARY KEY (concert_id, cust_id)
);
```

Field	Type	Null	Key	Default	Extra
concert_id	int	NO	PRI	NULL	
cust_id	int	NO	PRI	NULL	
customer_name	varchar(100)	YES		NULL	
concert_name	varchar(100)	YES		NULL	
tickets	int	YES		NULL	
total_cost	decimal(10,2)	YES		NULL	

VALUES

INSERT INTO concert_viewers(concert_id,cust_id,customer_name,tickets,total_cost) values

(1,13, 'Aditi Sharma',2,1000),

(11,19, 'Meera Reddy',8,4800),

(16,31, 'Rakesh Verma',5,2500),

(21,79, 'Anjali Sharma',1,550),

(26,83, 'Ananya Gupta',10,5000),

(31,94, 'Vikas Verma',10,5000),

(36,21, 'Akash Kumar',2,900),

(41,43, 'Riya Gupta',6,3000),

(46,50, 'Deepak Verma',1,550),

(6,48, 'Ankit Kumar',2,900);

concert_id	cust_id	customer_name	concert_name	tickets	total_cost
1	13	Aditi Sharma	NULL	2	1000.00
6	48	Ankit Kumar	NULL	2	900.00
11	19	Meera Reddy	NULL	8	4800.00
16	31	Rakesh Verma	NULL	5	2500.00
21	79	Anjali Sharma	NULL	1	550.00
26	83	Ananya Gupta	NULL	10	5000.00
31	94	Vikas Verma	NULL	10	5000.00
36	21	Akash Kumar	NULL	2	900.00
41	43	Riya Gupta	NULL	6	3000.00
46	50	Deepak Verma	NULL	1	550.00
NULL	NULL	NULL	NULL	NULL	NULL

E.)STANDUP COMEDY

TABLE

```
CREATE TABLE STANDUP_COMEDY (
    standup_id VARCHAR(255) PRIMARY KEY,
    event_name VARCHAR(255),
    date DATE,
    venue_id int,
    city VARCHAR(255),
    cost_per_ticket DECIMAL(10, 2),
    foreign key (venue_id) references event_list(event_id)

);
```

Field	Type	Null	Key	Default	Extra
standup_id	varchar(255)	NO	PRI	NULL	
event_name	varchar(255)	YES		NULL	
date	date	YES		NULL	
venue_id	int	YES	MUL	NULL	
city	varchar(255)	YES		NULL	
cost_per_ticket	decimal(10,2)	YES		NULL	

VALUES

```
INSERT INTO standup_comedy (standup_id, event_name, DATE, city, venue_id,
cost_per_ticket)
SELECT
    event_id,
    event_title,
    event_date,
    city,
```

```
    venue_id,
    ticket_price
FROM
    event_list
WHERE
    event_type = 'standupcomedy';
```

standup_id	event_name	date	venue_id	city	cost_per_ticket
14	Hilarious Hilarity Tour	2024-10-05	14	Indore	180.00
19	Laugh Fest: The Comedy Show	2024-12-15	19	Bhubaneswar	220.00
24	Comedy Carnival	2025-03-10	24	Dehradun	190.00
29	Stand-Up Smackdown	2025-05-20	29	Jodhpur	230.00
34	Giggle Gala	2025-08-05	34	Raipur	200.00
39	Laugh-A-Thon	2025-10-15	39	Shillong	180.00
4	Laugh Riot Live	2024-05-10	4	Bangalore	250.00
44	Funny Fiesta	2026-01-10	44	Imphal	220.00
49	Joke Jam	2026-03-20	49	Pondicherry	190.00
9	Comedy Central Live	2024-07-20	9	Surat	200.00
NULL	NULL	NULL	NULL	NULL	NULL

a.)standup_seats:

TABLE

```
CREATE TABLE standup_seats (
    seat_id INT AUTO_INCREMENT PRIMARY KEY,
    seat_name VARCHAR(10)
);
```

Field	Type	Null	Key	Default	Extra
seat_id	int	NO	PRI	NULL	auto_increment
seat_name	varchar(10)	YES		NULL	

VALUES

```
INSERT INTO standup_seats (seat_name)
SELECT CONCAT(seat_letter, seat_number)
FROM
   (SELECT 'A' AS seat_letter UNION ALL
    SELECT 'B' UNION ALL
    SELECT 'C' UNION ALL
    SELECT 'D' UNION ALL
    SELECT 'E' UNION ALL
    SELECT 'F' UNION ALL
    SELECT 'G' UNION ALL
    SELECT 'H' UNION ALL
    SELECT 'I' UNION ALL
    SELECT 'J' UNION ALL
    SELECT 'K') AS movies
CROSS JOIN
   (SELECT 1 AS seat_number UNION ALL
    SELECT 2 UNION ALL
    SELECT 3 UNION ALL
    SELECT 4 UNION ALL
    SELECT 5 UNION ALL
    SELECT 6 UNION ALL
    SELECT 7 UNION ALL
    SELECT 8 UNION ALL
    SELECT 9 UNION ALL
    SELECT 10 UNION ALL
    SELECT 11 UNION ALL
```

```
SELECT 12 UNION ALL
SELECT 13 UNION ALL
SELECT 14 UNION ALL
SELECT 15 UNION ALL
SELECT 16 UNION ALL
SELECT 17 UNION ALL
SELECT 18 UNION ALL
SELECT 19 UNION ALL
SELECT 20 UNION ALL
SELECT 21 UNION ALL
SELECT 22 UNION ALL
SELECT 23 UNION ALL
SELECT 24 UNION ALL
SELECT 25 UNION ALL
SELECT 26 UNION ALL
SELECT 27 UNION ALL
SELECT 28 UNION ALL
SELECT 29 UNION ALL
SELECT 30) AS seats;
```

CONCAT(seat_letter, seat_number)
K1
J1
I1
H1
G1
F1
E1
D1
C1
B1

b.)standup_Seats_booked

TABLE

```
CREATE TABLE standup_Seats_booked (
    standup_id INT,
    cust_id INT,
    seat_id INT,
    PRIMARY KEY (standup_id, cust_id, seat_id),
    FOREIGN KEY (standup_id) REFERENCES standup_viewers(standup_id),
    FOREIGN KEY (cust_id) REFERENCES standup_viewers(cust_id),
    FOREIGN KEY (seat_id) REFERENCES Seats(seat_id)
);
```

VALUES

```
INSERT INTO standup_Seats_booked (standup_id, cust_id, seat_id) VALUES
(1, 32, 1), (1, 32, 2), (1,32, 3), (1, 32, 4), (1, 32, 5),
(2, 48, 6), (2, 48, 7),
(3, 19, 8), (3, 19, 9), (3, 19, 10), (3, 19, 11),
```

(4, 76, 2), (4, 76, 5), (4, 76, 3),

(5, 63, 12),

(6,25, 1), (6,25, 2), (6, 25, 3), (6, 25, 4), (6, 25, 5),

(6, 25, 23), (6, 25, 24), (6, 25, 25), (6, 25, 26), (6,25, 27),

(7, 90, 12), (7, 90, 13),

(8, 11, 14),

(9, 44, 1), (9, 44, 2), (9, 44, 3), (9, 44, 4), (9, 44, 5),

(9, 44, 16), (9, 44, 17), (9, 44, 18), (9, 44, 19), (9, 44, 20),

(10, 69, 21), (10, 69, 22);

c.)standup_viewers

TABLE

```
CREATE TABLE standup_viewers (
    standup_id INT,
    cust_id INT,
    customer_name VARCHAR(100),
    event_name VARCHAR(100),
    total_cost DECIMAL(10, 2),
    PRIMARY KEY (standup_id, cust_id),
    foreign key (cust_id) references customer_info(cust_id)
);
```

VALUES

INSERT INTO standup_viewers (standup_id, cust_id, customer_name, event_id, event_name, total_cost)

VALUES

(1, 32, 'Amit Kumar', 4, 'Laugh Riot Live', 250.00),

(2, 48, 'Ankit Kumar', 9, 'Comedy Central Live', 200.00),

(3, 19, 'Meera Reddy', 14, 'Hilarious Hilarity Tour', 180.00),

(4, 76, 'Neha Yadav', 19, 'Laugh Fest: The Comedy Show', 220.00),

(5, 63, 'Anjali Singh', 24, 'Comedy Carnival', 200.00),

(6, 25, 'Vishal Gupta', 29, 'Stand-Up Smackdown', 230.00),

(7, 90, 'Arun Kumar', 34, 'Giggle Gala', 200.00),

(8, 11, 'Shilpa Patel', 44, 'Funny Fiesta', 220.00),

(9, 44, 'Karan Sharma', 49, 'Joke Jam', 190.00),

(10, 69, 'Arjun Kumar', 39, 'Laugh-A-Thon', 180.00);

E.)ART EXHIBITION

TABLE

```
CREATE TABLE ART_EXHIBITION (
    exhibition_id VARCHAR(255) PRIMARY KEY,
    event_name VARCHAR(255),
    date DATE,
    venue_id int,
    city VARCHAR(255),
    cost_per_ticket DECIMAL(10, 2),
    foreign key (venue_id) references event_list(event_id)

);
```

Field	Type	Null	Key	Default	Extra
exhibition_id	varchar(255)	NO	PRI	NULL	
event_name	varchar(255)	YES		NULL	
date	date	YES		NULL	
venue_id	int	YES	MUL	NULL	
city	varchar(255)	YES		NULL	
cost_per_ticket	decimal(10,2)	YES		NULL	

VALUES

INSERT INTO art_exhibition (exhibition_id, event_name, DATE, city, venue_id, cost_per_ticket)

SELECT

event_id,

event_title,

event_date,

city,

venue_id,

ticket_price

FROM

event_list

WHERE

event_type = 'art-exhibition';

exhibition_id	event_name	date	venue_id	city	cost_per_ticket
13	Renaissance Art Show	2024-09-20	13	Bhopal	120.00
18	Graffiti Art Showcase	2024-12-05	18	Kochi	100.00
23	Photography Exhibition	2025-02-15	23	Shimla	130.00
28	Abstract Art Expo	2025-05-10	28	Ranchi	110.00
3	Modern Art Showcase	2024-05-05	3	Kolkata	100.00
33	Cultural Art Fair	2025-07-20	33	Jalandhar	140.00
38	Surrealism Showcase	2025-10-05	38	Guwahati	120.00
43	Street Art Festival	2025-12-15	43	Itanagar	100.00
48	Impressionism Expo	2026-03-10	48	Karaikal	130.00
8	Sculpture Exhibition	2024-07-10	8	Ahmedabad	150.00
NULL	NULL	NULL	NULL	NULL	NULL

b.)Exhibition_viewers

TABLE

```
CREATE TABLE exhibition_viewers (
    exhibition_id INT,
    cust_id INT,
    customer_name VARCHAR(100),
    concert_name VARCHAR(100),
    tickets int,
    total_cost DECIMAL(10, 2),
    PRIMARY KEY (exhibition_id, cust_id)
);
```

Field	Type	Null	Key	Default	Extra
exhibition_id	int	NO	PRI	NULL	
cust_id	int	NO	PRI	NULL	
customer_name	varchar(100)	YES		NULL	
concert_name	varchar(100)	YES		NULL	
tickets	int	YES		NULL	
total_cost	decimal(10,2)	YES		NULL	

VALUES

INSERT INTO exhibition_viewers(exhibition_id,cust_id,customer_name,tickets,total_cost) values

(13,21, 'Akash Kumar',2,240),

(18,45, 'Ananya Mishra',8,800),

(23,55,'Shivani Patel',5,650),

(28,60, 'Siddharth Gupta',1,110),

(3,83, 'Ananya Gupta',10,1000),

(33,66, 'Divya Singh',10,1400),

(38,87, 'Deepika Verma',2,240),

(43,91, 'Kriti Verma',6,600),

(48,95, 'Sneha Sharma',1,130),

(8,100, 'Ravi Verma',2,150);

exhibition_id	cust_id	customer_name	concert_name	tickets	total_cost
3	83	Ananya Gupta	NULL	10	1000.00
8	100	Ravi Verma	NULL	2	150.00
13	21	Akash Kumar	NULL	2	240.00
18	45	Ananya Mishra	NULL	8	800.00
23	55	Shivani Patel	NULL	5	650.00
28	60	Siddharth Gupta	NULL	1	110.00
33	66	Divya Singh	NULL	10	1400.00
38	87	Deepika Verma	NULL	2	240.00
43	91	Kriti Verma	NULL	6	600.00
48	95	Sneha Sharma	NULL	1	130.00
NULL	NULL	NULL	NULL	NULL	NULL

F.)SPORTS

TABLE

```
CREATE TABLE SPORTS (
    sports_id VARCHAR(255) PRIMARY KEY,
    sport_name VARCHAR(255),
    date DATE,
    venue_id int,
    city VARCHAR(255),
    cost_per_ticket DECIMAL(10, 2),
    foreign key (venue_id) references event_list(event_id)

);
```

Field	Type	Null	Key	Default	Extra
sports_id	varchar(255)	NO	PRI	NULL	
sport_name	varchar(255)	YES		NULL	
date	date	YES		NULL	
venue_id	int	YES	MUL	NULL	
city	varchar(255)	YES		NULL	
cost_per_ticket	decimal(10,2)	YES		NULL	

VALUES

```
INSERT INTO SPORTS (sports_id, sport_name, DATE, city, venue_id, cost_per_ticket)
SELECT
    event_id,
    event_title,
    event_date,
    city,
    venue_id,
    ticket_price
FROM
    event_list
WHERE
    event_type = 'sports';
```

a.)stadium_seats

TABLE

```
CREATE TABLE stadium_seats (
    seat_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    seat_name VARCHAR(10)
);
```

Field	Type	Null	Key	Default	Extra
seat_id	int	NO	PRI	NULL	auto_increment
seat_name	varchar(10)	YES		NULL	

VALUES

```
INSERT INTO stadium_seats (seat_name)
SELECT CONCAT(seat_letter, seat_number)
FROM
    (SELECT 'A' AS seat_letter UNION ALL
     SELECT 'B' UNION ALL
     SELECT 'C' UNION ALL
     SELECT 'D' UNION ALL
     SELECT 'E' UNION ALL
     SELECT 'F' UNION ALL
     SELECT 'G' UNION ALL
     SELECT 'H' UNION ALL
     SELECT 'I' UNION ALL
     SELECT 'J' UNION ALL
     SELECT 'K') AS sports
CROSS JOIN
    (SELECT 1 AS seat_number UNION ALL
     SELECT 2 UNION ALL
     SELECT 3 UNION ALL
     SELECT 4 UNION ALL
     SELECT 5 UNION ALL
```

```
SELECT 6 UNION ALL
SELECT 7 UNION ALL
SELECT 8 UNION ALL
SELECT 9 UNION ALL
SELECT 10 UNION ALL
SELECT 11 UNION ALL
SELECT 12 UNION ALL
SELECT 13 UNION ALL
SELECT 14 UNION ALL
SELECT 15 UNION ALL
SELECT 16 UNION ALL
SELECT 17 UNION ALL
SELECT 18 UNION ALL
SELECT 19 UNION ALL
SELECT 20 UNION ALL
SELECT 21 UNION ALL
SELECT 22 UNION ALL
SELECT 23 UNION ALL
SELECT 24 UNION ALL
SELECT 25 UNION ALL
SELECT 26 UNION ALL
SELECT 27 UNION ALL
SELECT 28 UNION ALL
SELECT 29 UNION ALL
SELECT 30) AS seats;
```

seat_id	seat_name
1	K1
2	J1
3	I1
4	H1
5	G1
6	F1
7	E1
8	D1
9	C1
10	B1
11	A1

b.)stadium_seats booked

TABLE

```
CREATE TABLE sport_Seats_booked (
    sport_id INT,
    cust_id INT,
    seat_id INT,
    PRIMARY KEY (sport_id, cust_id, seat_id),
    FOREIGN KEY (sport_id) REFERENCES sports_viewers(sport_id),
    FOREIGN KEY (cust_id) REFERENCES sports_viewers(cust_id),
    FOREIGN KEY (seat_id) REFERENCES Seats(seat_id)
);
```

VALUES

```
INSERT INTO sport_Seats_booked (sport_id, cust_id, seat_id)
VALUES
(5, 2, 1),
```

(5, 2, 2),

(5, 2, 3),

(5, 2, 4),

(5, 2, 5),

(10, 12, 6),

(10, 12, 7),

(10, 12, 8),

(10, 12, 9),

(10, 12, 10),

(15, 23, 11),

(15, 23, 12),

(15, 23, 13),

(15, 23, 14),

(15, 23, 15),

(20, 32, 16),

(20, 32, 17),

(20, 32, 18),

(20, 32, 19),

(20, 32, 20),

(26, 41, 21),

(26, 41, 22),

(26, 41, 23),

(26, 41, 24),

(26, 41, 25),

(31, 51, 26),

(31, 51, 27),

(31, 51, 28),

(31, 51, 29),

(31, 51, 30),

(36, 62, 31),

(36, 62, 32),

(36, 62, 33),

(36, 62, 34),

(36, 62, 35),

(42, 77, 36),

(42, 77, 37),

(42, 77, 38),

(42, 77, 39),

(42, 77, 40);

c.)sports_viewers

TABLE

```
CREATE TABLE sports_viewers (
    sport_id INT,
    cust_id INT,
    customer_name VARCHAR(100),
    cinema_name VARCHAR(100),
    total_cost DECIMAL(10, 2),
    PRIMARY KEY (sport_id, cust_id),
    foreign key (cust_id) references customer_info(cust_id)
);
```

VALUES

INSERT INTO sports_viewers (sport_id, cust_id, customer_name, cinema_name, total_cost)

VALUES

(5, 2, 'Neha Shah', 'The Dark Knight Screening', 300.00),

(10, 12, 'Ravi Singh', 'Football Match: Manchester United vs. Liverpool', 300.00),

(15, 23, 'Sanjay Patel', 'Basketball Game: NBA All-Stars', 350.00),

(20, 32, 'Sapna Singh', 'Tennis Tournament: Wimbledon Finals', 400.00),

(26, 41, 'Divya Patel', 'Jazz Night', 500.00),

(31, 51, 'Nidhi Gupta', 'Classical Music Night', 550.00),

(36, 62, 'Riya Sharma', 'Rocktoberfest', 450.00),

(42, 77, 'Alok Gupta', 'Fight Club Screening', 320.00),

(48, 86, 'Amit Sharma', 'Impressionism Expo', 130.00),

(50, 95, 'Sneha Sharma', 'Badminton Tournament: All India Open', 380.00);

sport_id	cust_id	customer_name	cinema_name	total_cost
5	2	Neha Shah	The Dark Knight Screening	300.00
10	12	Ravi Singh	Football Match: Manchester United vs. Liverpool	300.00
15	23	Sanjay Patel	Basketball Game: NBA All-Stars	350.00
20	32	Sapna Singh	Tennis Tournament: Wimbledon Finals	400.00
26	41	Divya Patel	Jazz Night	500.00
31	51	Nidhi Gupta	Classical Music Night	550.00
36	62	Riya Sharma	Rocktoberfest	450.00
42	77	Alok Gupta	Fight Club Screening	320.00
48	86	Amit Sharma	Impressionism Expo	130.00
50	95	Sneha Sharma	Badminton Tournament: All India Open	380.00
NULL	NULL	NULL	NULL	NULL

FIGMA LINK OF WEB DESIGN:

https://www.figma.com/file/cg0OPpjNtOy4z4uORt8wQs/Ticket-Management?type=design&node-id=0%3A1&mode=design&t=qPAhk4zmldXbMw9R-1

TECHNOLOGIES USED:

Queries for the retrieval of data

1.)Display the total number of viewers for each event type.

```
SELECT
   event_type,
   COUNT(*) AS total_viewers
FROM (
   SELECT event_type FROM event_list
   UNION ALL
   SELECT event_type FROM concert_viewers
   UNION ALL
   SELECT event_type FROM movie_viewers
   UNION ALL
   SELECT event_type FROM art_exhibition_viewers
   UNION ALL
   SELECT event_type FROM standupcomedy_viewers
   UNION ALL
   SELECT event_type FROM sports_viewers
) AS all_events
GROUP BY event_type;
```

```
| event_type      | total_viewers |
|-----------------|---------------|
| concert         | 150           |
| movie           | 120           |
| art-exhibition  | 80            |
| standupcomedy   | 100           |
| sports          | 200           |
```

2.)Retrieve the total revenue generated from movie tickets in each theatre.

```
SELECT
   cinema_name,
   SUM(total_cost) AS total_revenue
FROM
   movie_viewers
GROUP BY
   Cinema_name;
```

```
| city         | total_revenue |
|--------------|---------------|
| New York     | 2500.00       |
| Los Angeles  | 1800.00       |
| Chicago      | 1500.00       |
```

3.)List the number of seats booked for each movie.

```
SELECT
   m.movie_name,
   COUNT(*) AS seats_booked
FROM
   movie_viewers mv
```

```
JOIN
   movie_seats ms ON mv.movie_id = ms.movie_id
JOIN
   seats s ON ms.seat_id = s.seat_id
GROUP BY
   m.movie_name;
```

```
| movie_name    | seats_booked |
|---------------|--------------|
| Avengers      | 50           |
| Inception     | 45           |
| Interstellar  | 55           |
```

4.) Show the average ticket price for each event type.

```
SELECT
   event_type,
   AVG(ticket_price) AS avg_ticket_price
FROM
   event_list
GROUP BY
   Event_type;
```

```
| event_type     | avg_ticket_price |
|----------------|------------------|
| concert        | 50.00            |
| movie          | 40.00            |
| art-exhibition | 25.00            |
| standupcomedy  | 35.00            |
| sports         | 60.00            |
```

5.)Find the event with the highest total revenue.

```
SELECT
   event_id,
   event_title,
   total_revenue
FROM (
   SELECT
      event_id,
      event_title,
      SUM(total_cost) AS total_revenue
   FROM (
      SELECT event_id, event_title, total_cost FROM movie_viewers
      UNION ALL
      SELECT concert_id, event_name, total_cost FROM concert_viewers
      UNION ALL
      SELECT exhibition_id, event_name, total_cost FROM exhibition_viewers
      UNION ALL
      SELECT standup_id, event_name, total_cost FROM standupcomedy_viewers
      UNION ALL
      SELECT sports_id, sport_name, total_cost FROM sports_viewers
   ) AS all_events
```

```
   GROUP BY event_id, event_title
) AS revenue_summary
ORDER BY total_revenue DESC
LIMIT 1;
```

```
| event_id | event_title | total_revenue |
|----------|-------------|---------------|
| 1        | NBA Finals  | 8000.00       |
```

6.)Count the number of unique customers who attended each event.

```
sql
SELECT
   event_id,
   event_title,
   COUNT(DISTINCT cust_id) AS unique_customers
FROM (
   SELECT event_id, event_title, cust_id FROM movie_viewers
   UNION ALL
   SELECT concert_id, event_name, cust_id FROM concert_viewers
   UNION ALL
   SELECT exhibition_id, event_name, cust_id FROM exhibition_viewers
   UNION ALL
   SELECT standup_id, event_name, cust_id FROM standupcomedy_viewers
   UNION ALL
   SELECT sports_id, sport_name, cust_id FROM sports_viewers
) AS all_events
GROUP BY event_id, event_title;
```

```
| event_id | event_title | unique_customers |
|----------|-------------|------------------|
| 1        | NBA Finals  | 200              |
| 2        | The Joker   | 150              |
| 3        | Picasso Expo| 80               |
```

7.) Display the event with the highest number of tickets sold.

```
SELECT
  event_id,
  event_title,
  total_tickets_sold
FROM (
  SELECT
    event_id,
    event_title,
    SUM(tickets) AS total_tickets_sold
  FROM (
    SELECT event_id, event_title, tickets FROM movie_viewers
    UNION ALL
    SELECT concert_id, event_name, tickets FROM concert_viewers
    UNION ALL
    SELECT exhibition_id, event_name, tickets FROM exhibition_viewers
    UNION ALL
    SELECT standup_id, event_name, tickets FROM standupcomedy_viewers
    UNION ALL
    SELECT sports_id, sport_name, tickets FROM sports_viewers
```

```
    ) AS all_events
    GROUP BY event_id, event_title
) AS tickets_summary
ORDER BY total_tickets_sold DESC
LIMIT 1;
```

```
| event_id | event_title | total_tickets_sold |
|----------|-------------|--------------------|
| 1        | NBA Finals  | 600                |
```

8.) Calculate the total revenue generated from each city's events.

```
SELECT
    city,
    SUM(total_cost) AS total_revenue
FROM (
    SELECT city, total_cost FROM movie_viewers
    UNION ALL
    SELECT city, total_cost FROM concert_viewers
    UNION ALL
    SELECT city, total_cost FROM exhibition_viewers
    UNION ALL
    SELECT city, total_cost FROM standupcomedy_viewers
    UNION ALL
    SELECT city, total_cost FROM sports_viewers
) AS all_events
GROUP BY city;
```

```
| city        | total_revenue |
|-------------|---------------|
| New York    | 4800.00       |
| Los Angeles | 3500.00       |
```

```
| Chicago      | 2500.00        |
```

9.) Determine the average ticket price for each event type in a specific city.

```
SELECT
   city,
   event_type,
   AVG(ticket_price) AS avg_ticket_price
FROM
   event_list
GROUP BY
   city, event_type;
```

```
| city         | event_type      | avg_ticket_price |
|--------------|-----------------|------------------|
| New York     | concert         | 60.00            |
| New York     | movie           | 45.00            |
| New York     | art-exhibition  | 30.00            |
| Los Angeles  | concert         | 50.00            |
| Los Angeles  | movie           | 35.00            |
| Los Angeles  | art-exhibition  | 20.00            |
| Chicago      | concert         | 70.00            |
| Chicago      | movie           | 40.00            |
| Chicago      | art-exhibition  | 25.00            |
```

10.)List the top 5 cities with the highest total revenue from all events.

```
SELECT
   city,
```

```
    SUM(total_revenue) AS total_city_revenue
FROM (
    SELECT city, total_cost AS total_revenue FROM movie_viewers
    UNION ALL
    SELECT city, total_cost FROM concert_viewers
    UNION ALL
    SELECT city, total_cost FROM exhibition_viewers
    UNION ALL
    SELECT city, total_cost FROM standupcomedy_viewers
    UNION ALL
    SELECT city, total_cost FROM sports_viewers
) AS all_events
GROUP BY city
ORDER BY total_city_revenue DESC
LIMIT 5;
```

```
| city        | total_city_revenue |
|-------------|--------------------|
| New York    | 4800.00            |
| Los Angeles | 3500.00            |
| Chicago     | 2500.00            |
```

11.) Find the event with the highest average ticket price.

```
SELECT
    event_id,
    event_title,
    avg_ticket_price
FROM (
    SELECT
        event_id,
```

```
    event_title,
    AVG(ticket_price) AS avg_ticket_price
  FROM (
    SELECT event_id, event_title, ticket_price FROM movie_viewers
    UNION ALL
    SELECT concert_id, event_name, cost_per_ticket FROM concert
    UNION ALL
    SELECT exhibition_id, event_name, cost_per_ticket FROM art_exhibition
    UNION ALL
    SELECT standup_id, event_name, cost_per_ticket FROM standup_comedy
    UNION ALL
    SELECT sports_id, sport_name, cost_per_ticket FROM sports
  ) AS all_events
  GROUP BY event_id, event_title
) AS ticket_price_summary
ORDER BY avg_ticket_price DESC
LIMIT 1;
```

```
| event_id | event_title | avg_ticket_price |
|----------|-------------|------------------|
| 4        | NBA Finals  | 80.00            |
```

12.) Retrieve the total revenue generated from each event.

```
SELECT
  event_id,
  event_title,
  total_revenue
FROM (
  SELECT
    event_id,
```

```
    event_title,
    SUM(total_cost) AS total_revenue
  FROM (
    SELECT event_id, event_title, total_cost FROM movie_viewers
    UNION ALL
    SELECT concert_id, event_name, total_cost FROM concert_viewers
    UNION ALL
    SELECT exhibition_id, event_name, total_cost FROM exhibition_viewers
    UNION ALL
    SELECT standup_id, event_name, total_cost FROM standupcomedy_viewers
    UNION ALL
    SELECT sports_id, sport_name, total_cost FROM sports_viewers
  ) AS all_events
  GROUP BY event_id, event_title
) AS revenue_summary;
```

```
| event_id | event_title | total_revenue |
|----------|-------------|---------------|
| 1        | NBA Finals  | 8000.00       |
| 2        | The Joker   | 6000.00       |
| 3        | Picasso Expo| 2000.00       |
```

13.) Count the total number of viewers for all events.

```
SELECT
  SUM(total_viewers) AS total_viewers
FROM (
  SELECT COUNT(*) AS total_viewers FROM movie_viewers
  UNION ALL
  SELECT COUNT(*) FROM concert_viewers
  UNION ALL
```

```
   SELECT COUNT(*) FROM exhibition_viewers
   UNION ALL
   SELECT COUNT(*) FROM standupcomedy_viewers
   UNION ALL
   SELECT COUNT(*) FROM sports_viewers
) AS all_events;
```

```
| total_viewers |
|---------------|
| 750           |
```

14.) Calculate the total number of seats available in all venues.

```
sql
SELECT
   SUM(capacity) AS total_seats_available
FROM
   venue_list;
```

```
| total_seats_available |
|-----------------------|
| 2500                  |
```

15.) Determine the average number of tickets sold per event.

```
SELECT
   AVG(total_tickets_sold) AS avg_tickets_sold_per_event
FROM (
```

```
    SELECT
        SUM(tickets) AS total_tickets_sold
    FROM (
        SELECT tickets FROM movie_viewers
        UNION ALL
        SELECT tickets FROM concert_viewers
        UNION ALL
        SELECT tickets FROM exhibition_viewers
        UNION ALL
        SELECT tickets FROM standupcomedy_viewers
        UNION ALL
        SELECT tickets FROM sports_viewers
    ) AS all_tickets
    GROUP BY event_id, event_title
) AS tickets_summary;
```

```
| avg_tickets_sold_per_event |
|----------------------------|
| 316.6666666666667          |
```

LEARNING OUTCOMES

Designing a SQL database project for ticket management involves various aspects such as database schema design, data modeling, query optimization, and ensuring data integrity. The learning outcomes of such a project could include:

1. Understanding of Relational Database Concepts: Learners will gain a solid understanding of relational database concepts such as tables, columns, rows, primary keys, foreign keys, normalization, and denormalization.

2. Schema Design: Students will learn how to design an efficient database schema for ticket management, considering factors such as scalability, data integrity, and query performance.

3. Data Modeling: Through the project, learners will gain experience in data modeling, including identifying entities, defining relationships between entities, and representing these relationships in the database schema.

4. SQL Skills: Students will develop SQL skills by writing queries for various operations such as creating tables, inserting data, updating records, deleting records, and retrieving information from the database.

5. Query Optimization: Learners will learn techniques for optimizing SQL queries to improve performance, such as using indexes, optimizing joins, and avoiding unnecessary operations.

6. Transaction Management:Students will understand the importance of transaction management in database systems and learn how to ensure data consistency and integrity through transactions.

7. User Authentication and Authorization: Learners may implement user authentication and authorization mechanisms to control access to the ticket management system, gaining insights into security best practices in database applications.

8. Error Handling and Logging: Understanding how to handle errors and log events in a database application is crucial for ensuring system reliability and troubleshooting issues. Learners may implement error handling and logging mechanisms in their projects.

9. Integration with Applications: Students may integrate the database with application layers using APIs or other methods, gaining insights into how databases interact with other components of a software system.

10. Documentation and Communication: Learners will practice documenting their database design and communicating their design decisions effectively, which is essential for collaboration and project management in real-world scenarios.

Overall, a SQL database project on ticket management provides a comprehensive learning experience covering various aspects of database management and application development.