ASSIGNMENT – 5
Submitted by : SANKAR ROY

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"

```
CREATE DATABASE TicketBookingSystem;
USE TicketBookingSystem;
```

2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
   - Venue

```
CREATE TABLE Venue (
    venue_id INT PRIMARY KEY,
    venue_name VARCHAR(255),
    address VARCHAR(255)
);

desc Venue;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
venue_id	int	NO	PRI	NULL	
venue_name	varchar(255)	YES		NULL	
address	varchar(255)	YES		NULL	

   - Event

```
CREATE TABLE Event (
    event_id INT PRIMARY KEY,
    event_name VARCHAR(255),
    event_date DATE,
    event_time TIME,
    venue_id INT,
    total_seats INT,
    available_seats INT,
    ticket_price DECIMAL,
    event_type  varchar(100) check(event_type in ('movie','sports','concert')),
    booking_id INT
    -- FOREIGN KEY (venue_id) REFERENCES Venue(venue_id),
    -- FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
event_id	int	NO	PRI	NULL	
event_name	varchar(255)	YES		NULL	
event_date	date	YES		NULL	
event_time	time	YES		NULL	
venue_id	int	YES	MUL	NULL	
total_seats	int	YES		NULL	
available_seats	int	YES		NULL	
ticket_price	decimal(10,0)	YES		NULL	
event_type	varchar(255)	YES		NULL	
booking_id	int	YES	MUL	NULL	

- Customers

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(15),
    booking_id INT
    -- FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);

desc Customer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
customer_name	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
phone_number	varchar(15)	YES		NULL	
booking_id	int	YES	MUL	NULL	

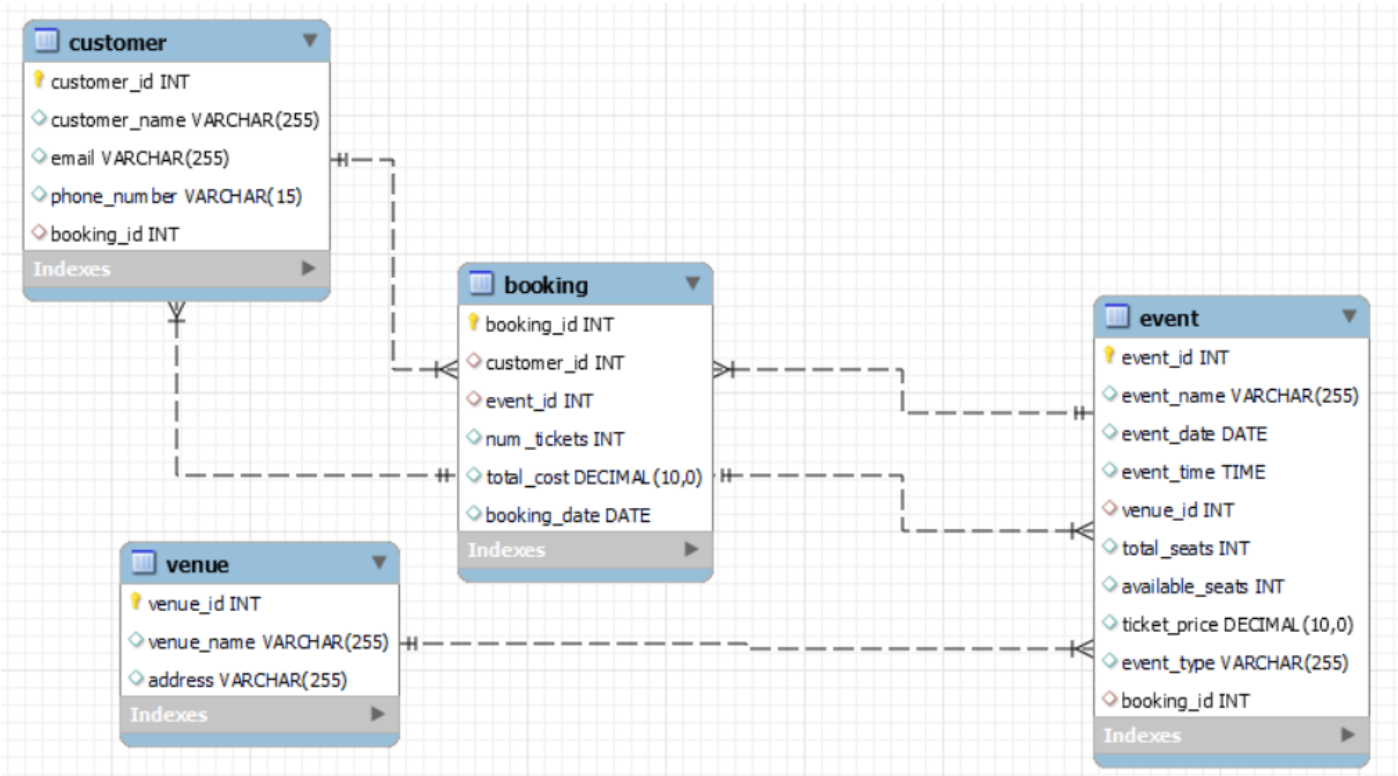
- Booking

```sql
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY,
    customer_id INT,
    event_id INT,
    num_tickets INT,
    total_cost DECIMAL,
    booking_date DATE
    -- FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
   -- FOREIGN KEY (event_id) REFERENCES Event(event_id)
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
booking_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
event_id	int	YES	MUL	NULL	
num_tickets	int	YES		NULL	
total_cost	decimal(10,0)	YES		NULL	
booking_date	date	YES		NULL	

3. Create an ERD (Entity Relationship Diagram) for the database.



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

```sql
alter table Event add constraint foreign key (venue_id) REFERENCES Venue(venue_id);
alter table Event add constraint foreign key (booking_id) REFERENCES Booking(booking_id);
alter table Customer add constraint foreign key (booking_id) REFERENCES Booking(booking_id);
alter table booking add constraint foreign key (customer_id) REFERENCES Customer(customer_id);
alter table booking add constraint foreign key (event_id) REFERENCES Event(event_id);
```

Tasks 2: Select, Where, Between, AND, LIKE:

1. Write a SQL query to insert at least 10 sample records into each table.

```sql
INSERT INTO Venue (venue_id, venue_name, address) VALUES
(1, 'Mumbai Exhibition Center', '123 Exhibition Road, Mumbai'),
(2, 'Delhi Sports Complex', '456 Sports Avenue, Delhi'),
(3, 'Chennai Concert Hall', '789 Music Street, Chennai'),
(4, 'Bangalore Movie Theater', '101 Film Lane, Bangalore'),
(5, 'Kolkata Convention Center', '202 Convention Road, Kolkata'),
(6, 'Hyderabad Stadium', '303 Sports Street, Hyderabad'),
(7, 'Jaipur Entertainment Plaza', '404 Entertainment Road, Jaipur'),
(8, 'Ahmedabad Arena', '505 Arena Lane, Ahmedabad'),
(9, 'Pune Cultural Center', '606 Cultural Street, Pune'),
(10, 'Lucknow Pavilion', '707 Pavilion Road, Lucknow');
```

venue_id	venue_name	address
1	Mumbai Exhibition Center	123 Exhibition Road, Mumbai
2	Delhi Sports Complex	456 Sports Avenue, Delhi
3	Chennai Concert Hall	789 Music Street, Chennai
4	Bangalore Movie Theater	101 Film Lane, Bangalore
5	Kolkata Convention Center	202 Convention Road, Kolkata
6	Hyderabad Stadium	303 Sports Street, Hyderabad
7	Jaipur Entertainment Plaza	404 Entertainment Road, Jaipur
8	Ahmedabad Arena	505 Arena Lane, Ahmedabad
9	Pune Cultural Center	606 Cultural Street, Pune
10	Lucknow Pavilion	707 Pavilion Road, Lucknow
NULL	NULL	NULL

```sql
INSERT INTO Event (event_id, event_name, event_date, event_time, total_seats, available_seats, ticket_price, event_type) VALUES
(1, 'Cricket Match', '2024-02-15', '18:00:00', 25000, 15000, 500, 'Sports'),
(2, 'Bollywood Night', '2024-03-10', '20:00:00',  5000, 3000, 1000, 'Concert'),
(3, 'Movie Premiere', '2024-04-05', '19:30:00',  1000, 800, 150, 'Movie'),
(4, 'Tech Expo', '2024-05-20', '10:00:00',  1500, 1200, 200, 'movie'),
(5, 'Kabaddi Championship', '2024-06-15', '17:30:00',  12000, 8000, 300, 'Sports'),
(6, 'Classical Music Concert', '2024-07-08', '19:00:00',  2000, 1500, 500, 'Concert'),
(7, 'Comedy Show', '2024-08-25', '21:00:00',  3000, 2500, 150, 'movie'),
(8, 'Drama Play', '2024-09-18', '18:30:00',  800, 600, 100, 'movie'),
(9, 'Football Match', '2024-10-12', '11:30:00',  5000, 4000, 50, 'sports'),
(10, 'Rock Band Concert', '2024-11-30', '22:00:00',  7000, 5000, 200, 'Concert');

select * from Event;
```

```sql
update event set booking_id = 1 where event_id = 1;
update event set booking_id = 2 where event_id = 2;
update event set booking_id = 3 where event_id = 3;
update event set booking_id = 4 where event_id = 4;
update event set booking_id = 5 where event_id = 5;
update event set booking_id = 6 where event_id = 6;
update event set booking_id = 7 where event_id = 7;
update event set booking_id = 8 where event_id = 8;
update event set booking_id = 9 where event_id = 9;
update event set booking_id = 10 where event_id = 10;
```

```sql
update event set venue_id = 2 where event_id = 1;
update event set venue_id = 3 where event_id = 2;
update event set venue_id = 4 where event_id = 3;
update event set venue_id = 1 where event_id = 4;
update event set venue_id = 6 where event_id = 5;
update event set venue_id = 5 where event_id = 6;
update event set venue_id = 7 where event_id = 7;
update event set venue_id = 9 where event_id = 8;
update event set venue_id = 8 where event_id = 9;
update event set venue_id = 10 where event_id = 10;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Cricket Match	2024-02-15	18:00:00	2	25000	15000	500	Sports	1
2	Bollywood Night	2024-03-10	20:00:00	3	5000	3000	1000	Concert	2
3	Movie Premiere	2024-04-05	19:30:00	4	1000	800	150	Movie	3
4	Tech Expo	2024-05-20	10:00:00	1	1500	1200	200	movie	4
5	Kabaddi Championship	2024-06-15	17:30:00	6	12000	8000	300	Sports	5
6	Classical Music Concert	2024-07-08	19:00:00	5	2000	1500	500	Concert	6
7	Comedy Show	2024-08-25	21:00:00	7	3000	2500	150	movie	7
8	Drama Play	2024-09-18	18:30:00	9	800	600	100	movie	8
9	football match	2024-10-12	11:30:00	8	5000	4000	50	sports	9
10	Rock Band Concert	2024-11-30	22:00:00	10	7000	5000	200	Concert	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```sql
INSERT INTO Customer (customer_id, customer_name, email, phone_number) VALUES
(1, 'Rajesh Kumar', 'rajesh@example.com', '9876543210'),
(2, 'Priya Singh', 'priya@example.com', '8765432109'),
(3, 'Amit Sharma', 'amit@example.com', '7654321098'),
(4, 'Sneha Verma', 'sneha@example.com', '6543210987'),
(5, 'Vikram Singh', 'vikram@example.com', '5432109876'),
(6, 'Pooja Gupta', 'pooja@example.com', '4321098765'),
(7, 'Rahul Kapoor', 'rahul@example.com', '3210987654'),
(8, 'Neha Singh', 'neha@example.com', '2109876543'),
(9, 'Sanjay Patel', 'sanjay@example.com', '1098765432'),
(10, 'Kavita Joshi', 'kavita@example.com', '9876543210');

select * from customer;
```

```sql
update customer set booking_id = 1 where customer_id = 1;
update customer set booking_id = 2 where customer_id = 2;
update customer set booking_id = 3 where customer_id = 3;
update customer set booking_id = 4 where customer_id = 4;
update customer set booking_id = 5 where customer_id = 5;
update customer set booking_id = 6 where customer_id = 6;
update customer set booking_id = 7 where customer_id = 7;
update customer set booking_id = 8 where customer_id = 8;
update customer set booking_id = 9 where customer_id = 9;
update customer set booking_id = 10 where customer_id = 10;
```

customer_id	customer_name	email	phone_number	booking_id
1	Rajesh Kumar	rajesh@example.com	9876543210	1
2	Priya Singh	priya@example.com	8765432109	2
3	Amit Sharma	amit@example.com	7654321098	3
4	Sneha Verma	sneha@example.com	6543210987	4
5	Vikram Singh	vikram@example.com	5432109876	5
6	Pooja Gupta	pooja@example.com	4321098765	6
7	Rahul Kapoor	rahul@example.com	3210987654	7
8	Neha Singh	neha@example.com	2109876543	8
9	Sanjay Patel	sanjay@example.com	1098765432	9
10	Kavita Joshi	kavita@example.com	9876543210	10
NULL	NULL	NULL	NULL	NULL

```sql
INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
(1, 1, 1, 5, 2500, '2024-02-10'),
(2, 2, 2, 3, 3000, '2024-03-05'),
(3, 3, 3, 2, 300, '2024-04-01'),
(4, 4, 5, 6, 1800, '2024-06-01'),
(5, 5, 6, 4, 2000, '2024-07-05'),
(6, 6, 7, 2, 300, '2024-08-20'),
(7, 7, 8, 1, 100, '2024-09-15'),
(8, 8, 10, 3, 600, '2024-11-25'),
(9, 9, 4, 10, 2000, '2024-05-15'),
(10, 10, 9, 5, 250, '2024-10-01');
```

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1	1	1	5	2500	2024-02-10
2	2	2	3	3000	2024-03-05
3	3	3	2	300	2024-04-01
4	4	5	6	1800	2024-06-01
5	5	6	4	2000	2024-07-05
6	6	7	2	300	2024-08-20
7	7	8	1	100	2024-09-15
8	8	10	3	600	2024-11-25
9	9	4	10	2000	2024-05-15
10	10	9	5	250	2024-10-01
NULL	NULL	NULL	NULL	NULL	NULL

2. Write a SQL query to list all Events.

```sql
select distinct * from  event;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Cricket Match	2024-02-15	18:00:00	2	25000	15000	500	Sports	1
2	Bollywood Night	2024-03-10	20:00:00	3	5000	3000	1000	Concert	2
3	Movie Premiere	2024-04-05	19:30:00	4	1000	800	150	Movie	3
4	Tech Expo	2024-05-20	10:00:00	1	1500	1200	200	movie	4
5	Kabaddi Championship	2024-06-15	17:30:00	6	12000	8000	300	Sports	5
6	Classical Music Concert	2024-07-08	19:00:00	5	2000	1500	500	Concert	6
7	Comedy Show	2024-08-25	21:00:00	7	3000	2500	150	movie	7
8	Drama Play	2024-09-18	18:30:00	9	800	600	100	movie	8
9	football match	2024-10-12	11:30:00	8	5000	4000	50	sports	9
10	Rock Band Concert	2024-11-30	22:00:00	10	7000	5000	200	Concert	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Write a SQL query to select events with available tickets.

```sql
SELECT * FROM Event WHERE available_seats > 0;
```

4. Write a SQL query to select events name partial match with ‘cup’.

```
154 •  select event_name from event where event_name like '%cup%';
155
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name

5. Write a SQL query to select events with ticket price range is between 1000 to 2500.

```
156 •  select * from event where ticket_price between 1000 and 2500;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
2	Bollywood Night	2024-03-10	20:00:00	3	5000	3000	1000	Concert	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6. Write a SQL query to retrieve events with dates falling within a specific range.

```
158 •  select * from event where event_date between '2024-05-20' and '2024-08-25';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
4	Tech Expo	2024-05-20	10:00:00	1	1500	1200	200	Conference	4
5	Kabaddi Championship	2024-06-15	17:30:00	6	12000	8000	300	Sports	5
6	Classical Music Concert	2024-07-08	19:00:00	5	2000	1500	500	Concert	6
7	Comedy Show	2024-08-25	21:00:00	7	3000	2500	150	Comedy	7
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name

```
160 •  SELECT * FROM Event WHERE available_seats > 0 AND event_name like '%Concert';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
6	Classical Music Concert	2024-07-08	19:00:00	5	2000	1500	500	Concert	6
10	Rock Band Concert	2024-11-30	22:00:00	10	7000	5000	200	Concert	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.

```
161
162 •   SELECT * FROM Customer LIMIT 5 OFFSET 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

customer_id	customer_name	email	phone_number	booking_id
6	Pooja Gupta	pooja@example.com	4321098765	6
7	Rahul Kapoor	rahul@example.com	3210987654	7
8	Neha Singh	neha@example.com	2109876543	8
9	Sanjay Patel	sanjay@example.com	1098765432	9
10	Kavita Joshi	kavita@example.com	9876543210	10
NULL	NULL	NULL	NULL	NULL

9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.

```
164 •   select * from booking where num_tickets >4;
```

Result Grid | Filter Rows: | Edit: | Export/Impor

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1	1	1	5	2500	2024-02-10
4	4	5	6	1800	2024-06-01
9	9	4	10	2000	2024-05-15
10	10	9	5	250	2024-10-01
NULL	NULL	NULL	NULL	NULL	NULL

10. Write a SQL query to retrieve customer information whose phone number end with '000'

```
select * from customer where phone_number = "%000";
```

Result Grid | Filter Rows: | Edit: | Export/I

customer_id	customer_name	email	phone_number	booking_id
NULL	NULL	NULL	NULL	NULL

11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.

```
select * from event having total_seats > 15000
order by total_seats;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Cricket Match	2024-02-15	18:00:00	2	25000	15000	500	Sports	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

12. Write a SQL query to select events name not start with 'x', 'y', 'z'

```
SELECT * FROM Event WHERE event_name NOT LIKE 'x%' or event_name NOT LIKE 'y%' or event_name NOT LIKE 'z%';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Cricket Match	2024-02-15	18:00:00	2	25000	15000	500	Sports	1
2	Bollywood Night	2024-03-10	20:00:00	3	5000	3000	1000	Concert	2
3	Movie Premiere	2024-04-05	19:30:00	4	1000	800	150	Movie	3
4	Tech Expo	2024-05-20	10:00:00	1	1500	1200	200	movie	4
5	Kabaddi Championship	2024-06-15	17:30:00	6	12000	8000	300	Sports	5
6	Classical Music Concert	2024-07-08	19:00:00	5	2000	1500	500	Concert	6
7	Comedy Show	2024-08-25	21:00:00	7	3000	2500	150	movie	7
8	Drama Play	2024-09-18	18:30:00	9	800	600	100	movie	8
9	football match	2024-10-12	11:30:00	8	5000	4000	50	sports	9
10	Rock Band Concert	2024-11-30	22:00:00	10	7000	5000	200	Concert	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write a SQL query to List Events and Their Average Ticket Prices.

```sql
SELECT event_name, AVG(ticket_price) AS average_ticket_price
FROM Event
GROUP BY event_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name	average_ticket_price
Cricket Match	500.0000
Bollywood Night	1000.0000
Movie Premiere	150.0000
Tech Expo	200.0000
Kabaddi Championship	300.0000
Classical Music Concert	500.0000
Comedy Show	150.0000
Drama Play	100.0000
football match	50.0000
Rock Band Concert	200.0000

2. Write a SQL query to Calculate the Total Revenue Generated by Events.

```sql
SELECT event_name, SUM(total_cost) AS total_revenue
FROM Booking
JOIN Event ON Booking.event_id = Event.event_id
GROUP BY event_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

event_name	total_revenue
Cricket Match	2500
Bollywood Night	3000
Movie Premiere	300
Kabaddi Championship	1800
Classical Music Concert	2000
Comedy Show	300
Drama Play	100
Rock Band Concert	600
Tech Expo	2000
football match	250

3. Write a SQL query to find the event with the highest ticket sales.

```sql
select event.event_name , sum(num_tickets) as ticketSold
from booking
join event on booking.event_id = event.event_id
group by event_name
order by ticketSold desc
limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name	ticketSold
Tech Expo	10

4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.

```sql
select event.event_name , sum(num_tickets) as ticketSold
from booking
join event on booking.event_id = event.event_id
group by event_name
order by ticketSold;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name	ticketSold
Drama Play	1
Movie Premiere	2
Comedy Show	2
Bollywood Night	3
Rock Band Concert	3
Classical Music Concert	4
Cricket Match	5
football match	5
Kabaddi Championship	6
Tech Expo	10

5. Write a SQL query to Find Events with No Ticket Sales.

```sql
select event.event_name
from booking
join event on event.event_id = booking.event_id
where num_tickets is null
group by event_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

event_name

6. Write a SQL query to Find the User Who Has Booked the Most Tickets.

```sql
select customer.customer_name , sum(num_tickets) as TicketBooked
from booking
join customer on customer.customer_id = booking.customer_id
group by customer_name
order by TicketBooked desc
limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch ro

customer_name	TicketBooked
Sanjay Patel	10

7. Write a SQL query to List Events and the total number of tickets sold for each month.

```sql
SELECT event.event_name , DATE_FORMAT(booking.booking_date, '%Y-%m') AS month, count(booking.booking_id) AS total_tickets_sold
FROM Booking
join event on event.event_id = booking.event_id
GROUP BY booking.booking_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name	month	total_tickets_sold
Cricket Match	2024-02	1
Bollywood Night	2024-03	1
Movie Premiere	2024-04	1
Kabaddi Championship	2024-06	1
Classical Music Concert	2024-07	1
Comedy Show	2024-08	1
Drama Play	2024-09	1
Rock Band Concert	2024-11	1
Tech Expo	2024-05	1
football match	2024-10	1

8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.

```sql
select event.event_name , avg(ticket_price) as avgPrice
from venue
join event on venue.venue_id = event.venue_id
group by event_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name	avgPrice
Tech Expo	200.0000
Cricket Match	500.0000
Bollywood Night	1000.0000
Movie Premiere	150.0000
Classical Music Concert	500.0000
Kabaddi Championship	300.0000
Comedy Show	150.0000
football match	50.0000
Drama Play	100.0000
Rock Band Concert	200.0000

9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.

```sql
SELECT event_type, SUM(num_tickets) AS total_tickets_sold
FROM Booking
JOIN Event ON Booking.event_id = Event.event_id
GROUP BY event_type;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_type	total_tickets_sold
Sports	16
Concert	10
Movie	15

10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.

```sql
SELECT YEAR(booking_date) AS year, SUM(total_cost) AS total_revenue
FROM Booking
GROUP BY year;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	total_revenue
2024	12850

11. Write a SQL query to list users who have booked tickets for multiple events.

```sql
SELECT customer_name, COUNT(DISTINCT Booking.event_id) AS events_booked
FROM Customer
JOIN Booking ON Customer.customer_id = Booking.customer_id
GROUP BY customer_name
HAVING events_booked > 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	events_booked

12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.

```sql
SELECT customer_name, SUM(total_cost) AS total_revenue
FROM Customer
JOIN Booking ON Customer.customer_id = Booking.customer_id
GROUP BY customer_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	total_revenue
Rajesh Kumar	2500
Priya Singh	3000
Amit Sharma	300
Sneha Verma	1800
Vikram Singh	2000
Pooja Gupta	300
Rahul Kapoor	100
Neha Singh	600
Sanjay Patel	2000
Kavita Joshi	250

13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.

```sql
SELECT venue_id, event_type, AVG(ticket_price) AS average_ticket_price
FROM Event
GROUP BY venue_id, event_type;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

venue_id	event_type	average_ticket_price
2	Sports	500.0000
3	Concert	1000.0000
4	Movie	150.0000
1	movie	200.0000
6	Sports	300.0000
5	Concert	500.0000
7	movie	150.0000
9	movie	100.0000
8	sports	50.0000
10	Concert	200.0000

14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30Days.

```sql
SELECT customer_name, SUM(num_tickets) AS total_tickets_purchased
FROM Customer
JOIN Booking ON Customer.customer_id = Booking.customer_id
WHERE booking_date >= CURDATE() - INTERVAL 30 DAY
GROUP BY customer_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	total_tickets_purchased
Rajesh Kumar	5
Priya Singh	3
Amit Sharma	2
Sneha Verma	6
Vikram Singh	4
Pooja Gupta	2
Rahul Kapoor	1
Neha Singh	3
Sanjay Patel	10
Kavita Joshi	5

Tasks 4: Subquery and its types

1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.

```sql
SELECT venue_id, AVG(ticket_price) AS average_ticket_price
FROM Event
WHERE venue_id IN (SELECT DISTINCT venue_id FROM Event)
GROUP BY venue_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

venue_id	average_ticket_price
1	200.0000
2	500.0000
3	1000.0000
4	150.0000
5	500.0000
6	300.0000
7	150.0000
8	50.0000
9	100.0000
10	200.0000

2. Find Events with More Than 50% of Tickets Sold using subquery.

```sql
SELECT event_name
FROM Event
WHERE event_id IN (
    SELECT event_id
    FROM Booking
    GROUP BY event_id
    HAVING SUM(num_tickets) > 0.5 * total_seats
);
```

Result Grid | Filter Rows: | Export: | Wrap C

event_name

3. Calculate the Total Number of Tickets Sold for Each Event.

```
SELECT event_name,
    (SELECT SUM(num_tickets) FROM Booking WHERE Event.event_id = Booking.event_id) AS total_tickets_sold
FROM Event;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name	total_tickets_sold
Cricket Match	5
Bollywood Night	3
Movie Premiere	2
Tech Expo	10
Kabaddi Championship	6
Classical Music Concert	4
Comedy Show	2
Drama Play	1
football match	5
Rock Band Concert	3

4. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.

```
SELECT customer_name
FROM Customer
WHERE NOT EXISTS (SELECT 1 FROM Booking WHERE Customer.customer_id = Booking.customer_id);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name

5. List Events with No Ticket Sales Using a NOT IN Subquery.

```
SELECT event_name
FROM Event
WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name

6. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROMClause.

```sql
SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
FROM (
    SELECT event_type, COUNT(event.booking_id) AS total_tickets_sold
    FROM Event
    LEFT JOIN Booking ON Event.event_id = Booking.event_id
    GROUP BY event_type, Event.event_id
) AS subquery
GROUP BY event_type;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_type	total_tickets_sold
Sports	3
Concert	3
Movie	4

7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in theWHERE Clause.

```sql
SELECT event_name
FROM Event
WHERE ticket_price > (SELECT AVG(ticket_price) FROM Event);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_name
Cricket Match
Bollywood Night
Classical Music Concert

8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.

```sql
SELECT customer_name,
    (SELECT SUM(total_cost) FROM Booking WHERE Customer.customer_id = Booking.customer_id) AS total_revenue
FROM Customer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	total_revenue
Rajesh Kumar	2500
Priya Singh	3000
Amit Sharma	300
Sneha Verma	1800
Vikram Singh	2000
Pooja Gupta	300
Rahul Kapoor	100
Neha Singh	600
Sanjay Patel	2000
Kavita Joshi	250

9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHEREClause.

```sql
SELECT customer_name
FROM Customer
WHERE customer_id IN (SELECT DISTINCT customer_id FROM Booking WHERE event_id IN (SELECT event_id FROM Event WHERE venue_id = 1));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name
Sanjay Patel

10. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery withGROUP BY.

```sql
SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
FROM (
    SELECT event_type, COUNT(event.booking_id) AS total_tickets_sold
    FROM Event
    LEFT JOIN Booking ON Event.event_id = Booking.event_id
    GROUP BY event_type, Event.event_id
) AS subquery
GROUP BY event_type;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

event_type	total_tickets_sold
Sports	3
Concert	3
Movie	4

11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery withDATE_FORMAT.

```
SELECT customer_name, DATE_FORMAT(booking_date, '%Y-%m') AS month
FROM Customer
JOIN Booking ON Customer.customer_id = Booking.customer_id
GROUP BY customer_name, month;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	month
Rajesh Kumar	2024-02
Priya Singh	2024-03
Amit Sharma	2024-04
Sneha Verma	2024-06
Vikram Singh	2024-07
Pooja Gupta	2024-08
Rahul Kapoor	2024-09
Neha Singh	2024-11
Sanjay Patel	2024-05
Kavita Joshi	2024-10

12. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery

```
SELECT venue_id, AVG(ticket_price) AS average_ticket_price
FROM Event
WHERE venue_id IN (SELECT DISTINCT venue_id FROM Event)
GROUP BY venue_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

venue_id	average_ticket_price
1	200.0000
2	500.0000
3	1000.0000
4	150.0000
5	500.0000
6	300.0000
7	150.0000
8	50.0000
9	100.0000
10	200.0000