HEXAWARE ASSIGNMENT 1

Ticket Booking System

You are tasked with creating a ticket booking system for a Event. The system should support booking tickets for different types of events, such as movies, concerts, and plays. Each event has its own pricing strategy, and the system should also track available seats and customer bookings.

Database Tables

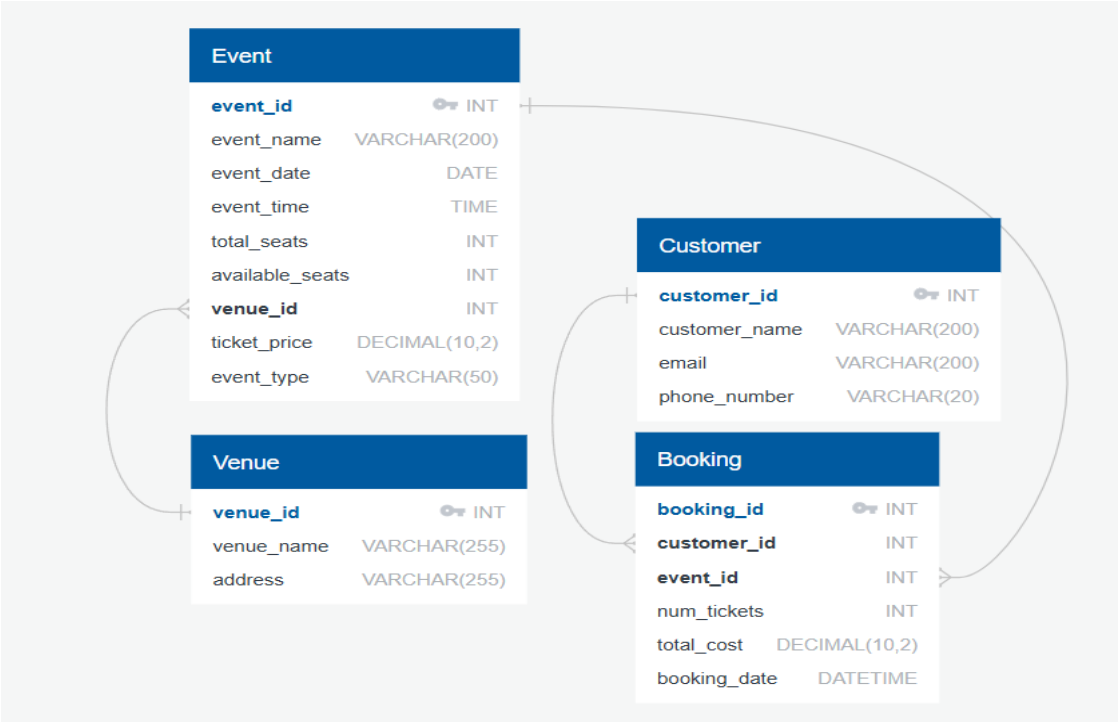
1. Venue Table
    - venue_id (Primary Key)
    - venue_name,
    - address
2. Event Table
    - event_id (Primary Key)
    - event_name,
    - event_date DATE,
    - event_time TIME,
    - venue_id (Foreign Key),
    - total_seats,
    - available_seats,
    - ticket_price DECIMAL,
    - event_type ('Movie', 'Sports', 'Concert')
    - booking_id (Foreign Key).
3. Customer Table
    - customer_id (Primary key)
    - customer_name,
    - email,
    - phone_number,
    - booking_id (Foreign Key)
4. Booking Table
    1. booking_id (Primary Key),
    2. customer_id (Foreign Key),
    3. event_id (Foreign Key),
    4. num_tickets,
    5. total_cost,
    6. booking_date,

Tasks 1: Database Design:

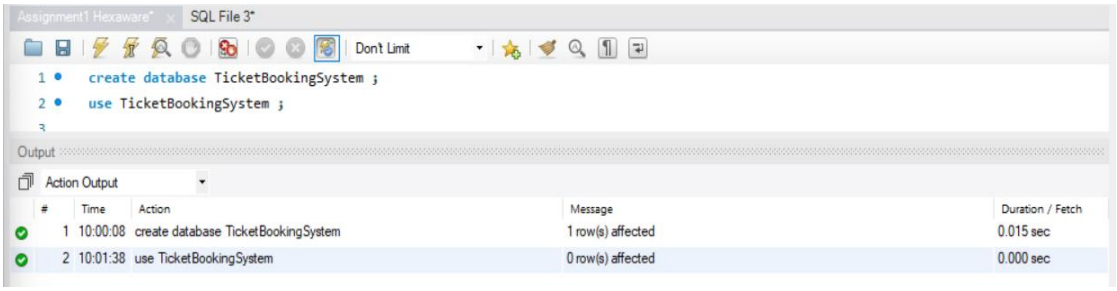
1. Create the database named "TicketBookingSystem"
2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
    - Venue

- Event
- Customers
- Booking

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



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



```sql
-- create venue table
Create table Venue(
    venue_id int primary key auto_increment,
    venue_name varchar(255) not null,
    address varchar(255) not null
);
-- create event table
Create table Event(
    event_id int primary key auto_increment,
    event_name varchar(200) not null,
    event_date DATE not null,
    event_time TIME not null,
    total_seats int not null,
    available_seats int not null,
    venue_id int not null,
    ticket_price DECIMAL(10,2) not null,
    event_type enum('Movie', 'Sports', 'Concert'),
    foreign Key (venue_id) references Venue(venue_id)
    );
-- create customer table
Create table Customer (
    customer_id int primary key auto_increment,
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
3	10:34:19	Create table Venue(venue_id int primary key auto_increment, venue_na...	0 row(s) affected	0.031 sec
4	10:34:26	Create table Event(event_id int primary key auto_increment, event_na...	0 row(s) affected	0.063 sec

```sql
-- create customer table
Create table Customer (
    customer_id int primary key auto_increment,
    customer_name varchar(200) not null,
    email varchar(200) not null,
    phone_number int not null
    );
-- create booking table
Create table Booking (
    booking_id int primary key auto_increment,
    customer_id int not null,
    event_id int not null,
    num_tickets int not null,
    total_cost decimal(10,2) not null,
    booking_date datetime default current_timestamp,
    foreign key (customer_id) references Customer (customer_id),
    foreign key (event_id) references Event (event_id)
    );
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:53:58	create database TicketBookingSystem	1 row(s) affected	0.016 sec
2	10:54:07	use TicketBookingSystem	0 row(s) affected	0.000 sec
3	10:54:18	Create table Venue(venue_id int primary key auto_increment, venue_nam...	0 row(s) affected	0.032 sec
4	10:54:22	Create table Event(event_id int primary key auto_increment, event_nam...	0 row(s) affected	0.062 sec
5	10:54:30	Create table Customer (customer_id int primary key auto_increment, cust...	0 row(s) affected	0.047 sec
6	10:54:35	Create table Booking (booking_id int primary key auto_increment, custo...	0 row(s) affected	0.063 sec

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

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

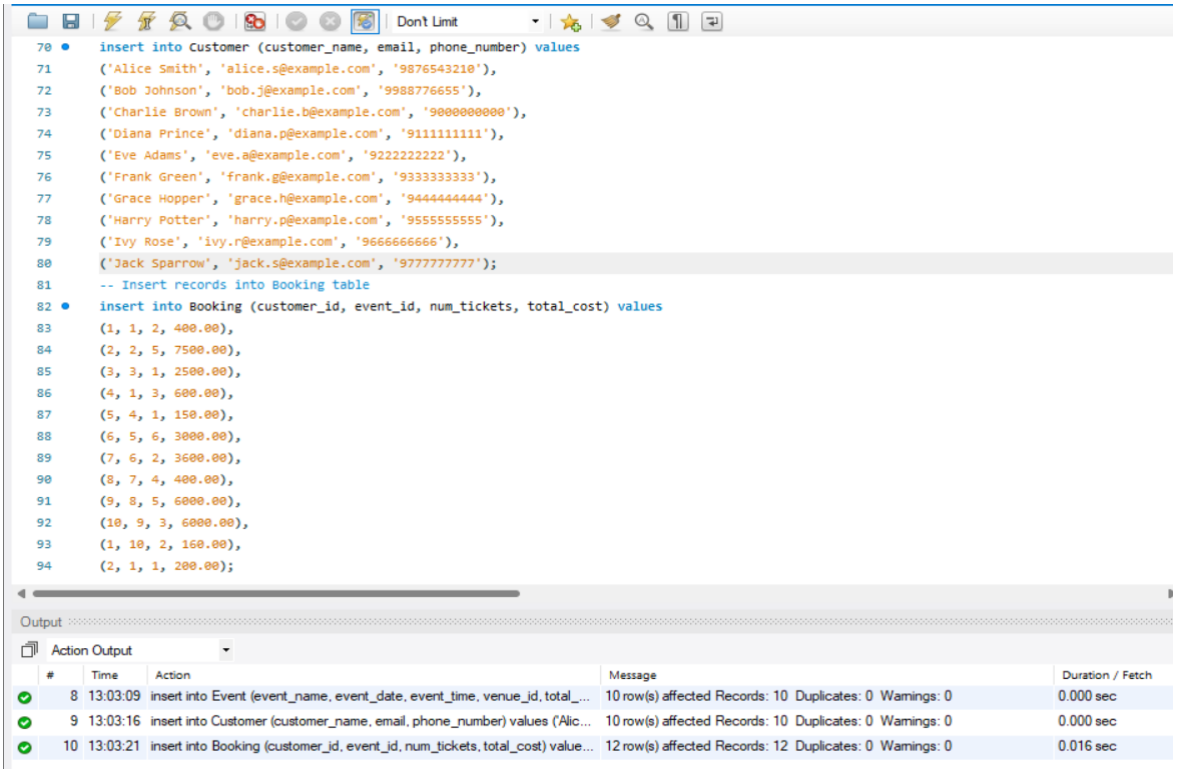
```sql
-- Insert records into Venue table
insert into Venue (venue_name, address) values
('Grand Cinema', '123 Movie St, CityA'),
('City Stadium', '456 Sports Rd, CityB'),
('Concert Hall', '789 Music Ave, CityC'),
('Exhibition Center', '101 Expo Blvd, CityD'),
('Community Theater', '202 Play Ln, CityE'),
('Museum of Art', '303 Gallery Pkwy, CityF'),
('Science Center', '404 Discovery Dr, CityG'),
('Botanical Gardens', '505 Green St, CityH'),
('Waterfront Park', '606 Oceanfront Rd, CityI'),
('Innovation Hub', '707 Tech Ave, CityJ');
-- Insert records into Event table
insert into Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type) VALUES
('The Dark Knight Rises', '2025-07-15', '19:00:00', 1, 300, 250, 200.00, 'Movie'),
('Cricket World Cup Final', '2025-08-20', '14:30:00', 2, 50000, 48000, 1500.00, 'Sports'),
('Rock Legends Concert', '2025-09-10', '20:00:00', 3, 5000, 4500, 2500.00, 'Concert'),
('Comedy Night Live', '2025-07-25', '21:00:00', 4, 150, 100, 150.00, 'Movie'),
('Annual Sports Meet', '2025-08-01', '09:00:00', 5, 10000, 9000, 500.00, 'Sports'),
('Classical Music Gala', '2025-09-22', '18:00:00', 3, 800, 750, 1800.00, 'Concert'),
('Summer Movie Marathon', '2025-07-20', '10:00:00', 1, 400, 350, 100.00, 'Movie'),
('Football Championship', '2025-10-05', '16:00:00', 2, 30000, 28000, 1200.00, 'Sports'),
('Jazz Fusion Night', '2025-10-15', '20:30:00', 6, 600, 550, 2000.00, 'Concert'),
('Kids Movie Festival', '2025-07-18', '11:00:00', 7, 200, 180, 80.00, 'Movie');

-- Insert records into Customer table
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	11:35:07	insert into Venue (venue_name, address) values ('Grand Cinema', '123 M...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.094 sec
2	11:36:07	insert into Event (event_name, event_date, event_time, venue_id, total_...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.016 sec



```sql
insert into Customer (customer_name, email, phone_number) values
('Alice Smith', 'alice.s@example.com', '9876543210'),
('Bob Johnson', 'bob.j@example.com', '9988776655'),
('Charlie Brown', 'charlie.b@example.com', '9000000000'),
('Diana Prince', 'diana.p@example.com', '9111111111'),
('Eve Adams', 'eve.a@example.com', '9222222222'),
('Frank Green', 'frank.g@example.com', '9333333333'),
('Grace Hopper', 'grace.h@example.com', '9444444444'),
('Harry Potter', 'harry.p@example.com', '9555555555'),
('Ivy Rose', 'ivy.r@example.com', '9666666666'),
('Jack Sparrow', 'jack.s@example.com', '9777777777');
-- Insert records into Booking table
insert into Booking (customer_id, event_id, num_tickets, total_cost) values
(1, 1, 2, 400.00),
(2, 2, 5, 7500.00),
(3, 3, 1, 2500.00),
(4, 1, 3, 600.00),
(5, 4, 1, 150.00),
(6, 5, 6, 3000.00),
(7, 6, 2, 3600.00),
(8, 7, 4, 400.00),
(9, 8, 5, 6000.00),
(10, 9, 3, 6000.00),
(1, 10, 2, 160.00),
(2, 1, 1, 200.00);
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
8	13:03:09	insert into Event (event_name, event_date, event_time, venue_id, total_...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
9	13:03:16	insert into Customer (customer_name, email, phone_number) values ('Alic...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
10	13:03:21	insert into Booking (customer_id, event_id, num_tickets, total_cost) value...	12 row(s) affected Records: 12 Duplicates: 0 Warnings: 0	0.016 sec

2. Write a SQL query to list all Events.

```
101      -- TASK 2
102      --  listing all Events
103 •    select * from event;
104
```

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

event_id	event_name	event_date	event_time	total_seats	available_seats	venue_id	ticket_price	event_type
1	The Dark Knight Rises	2025-07-15	19:00:00	300	250	1	200.00	Movie
2	Unbooked Test Movie	2025-07-30	17:00:00	100	100	1	250.00	Movie
3	Cricket World Cup Final	2025-08-20	14:30:00	50000	48000	2	1500.00	Sports
4	Rock Legends Concert	2025-09-10	20:00:00	5000	4500	3	2500.00	Concert
5	Comedy Night Live	2025-07-25	21:00:00	150	100	4	150.00	Movie
6	Annual Sports Meet	2025-08-01	09:00:00	10000	9000	5	500.00	Sports
7	Classical Music Gala	2025-09-22	18:00:00	800	750	3	1800.00	Concert
8	Summer Movie Marathon	2025-07-20	10:00:00	400	350	1	100.00	Movie
9	Football Championship	2025-10-05	16:00:00	30000	28000	2	1200.00	Sports
10	Jazz Fusion Night	2025-10-15	20:30:00	600	550	6	2000.00	Concert
11	AI fusion	2025-06-20	10:00:00	100	40	1	500.00	Concert
12	Kids Movie Festival	2025-07-18	11:00:00	200	180	7	80.00	Movie

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

```
104
105      -- events with available tickets
106 •    select * from event where available_seats > 0 ;
107
```

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

event_id	event_name	event_date	event_time	total_seats	available_seats	venue_id	ticket_price	event_type
1	The Dark Knight Rises	2025-07-15	19:00:00	300	250	1	200.00	Movie
2	Unbooked Test Movie	2025-07-30	17:00:00	100	100	1	250.00	Movie
3	Cricket World Cup Final	2025-08-20	14:30:00	50000	48000	2	1500.00	Sports
4	Rock Legends Concert	2025-09-10	20:00:00	5000	4500	3	2500.00	Concert
5	Comedy Night Live	2025-07-25	21:00:00	150	100	4	150.00	Movie
6	Annual Sports Meet	2025-08-01	09:00:00	10000	9000	5	500.00	Sports
7	Classical Music Gala	2025-09-22	18:00:00	800	750	3	1800.00	Concert
8	Summer Movie Marathon	2025-07-20	10:00:00	400	350	1	100.00	Movie
9	Football Championship	2025-10-05	16:00:00	30000	28000	2	1200.00	Sports
10	Jazz Fusion Night	2025-10-15	20:30:00	600	550	6	2000.00	Concert
11	AI fusion	2025-06-20	10:00:00	100	40	1	500.00	Concert
12	Kids Movie Festival	2025-07-18	11:00:00	200	180	7	80.00	Movie

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

```
101
102      --  events name partial match with 'cup'
103 •    select * from event where event_name like '%cup%';
104
```

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

event_id	event_name	event_date	event_time	total_seats	available_seats	venue_id	ticket_price	event_type
2	Cricket World Cup Final	2025-08-20	14:30:00	50000	48000	2	1500.00	Sports

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

```
104
105     -- events with ticket price range is between 1000 to 2500
106 •   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	total_seats	available_seats	venue_id	ticket_price	event_type
2	Cricket World Cup Final	2025-08-20	14:30:00	50000	48000	2	1500.00	Sports
3	Rock Legends Concert	2025-09-10	20:00:00	5000	4500	3	2500.00	Concert
6	Classical Music Gala	2025-09-22	18:00:00	800	750	3	1800.00	Concert
8	Football Championship	2025-10-05	16:00:00	30000	28000	2	1200.00	Sports
9	Jazz Fusion Night	2025-10-15	20:30:00	600	550	6	2000.00	Concert

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

```
107
108     -- events with dates falling within a specific range
109     select * from event where event_date between '2025-07-10' and '2025-07-15';
110
```

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

event_id	event_name	event_date	event_time	total_seats	available_seats	venue_id	ticket_price	event_type
1	The Dark Knight Rises	2025-07-15	19:00:00	300	250	1	200.00	Movie

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

```
111     -- events with available tickets that also have "Concert" in their name
112 •   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	total_seats	available_seats	venue_id	ticket_price	event_type
3	Rock Legends Concert	2025-09-10	20:00:00	5000	4500	3	2500.00	Concert

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

```
113
114     -- users in batches of 5, starting from the 6th user
115 •   select * from customer limit 5 offset 5;
```

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

customer_id	customer_name	email	phone_number
6	Frank Green	frank.g@example.com	9333333333
7	Grace Hopper	grace.h@example.com	9444444444
8	Harry Potter	harry.p@example.com	9555555555
9	Ivy Rose	ivy.r@example.com	9666666666
10	Jack Sparrow	jack.s@example.com	9777777777

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

```
116
117     -- bookings details contains booked no of ticket more than 4
118 •   select * from Booking where num_tickets > 4 ;
```

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

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
2	2	2	5	7500.00	2025-06-11 14:43:38
6	6	5	6	3000.00	2025-06-11 14:43:38
9	9	8	5	6000.00	2025-06-11 14:43:38

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

```
119
120     -- customer information whose phone number end with '000'.
121 •   select * from Customer where phone_number like '%000';
122
```

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

customer_id	customer_name	email	phone_number
3	Charlie Brown	charlie.b@example.com	9000000000

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

```
122
123     -- events in order whose seat capacity more than 15000
124 •   select * from event where total_seats > 15000 order by total_seats ;
125
```

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

event_id	event_name	event_date	event_time	total_seats	available_seats	venue_id	ticket_price	event_type
8	Football Championship	2025-10-05	16:00:00	30000	28000	2	1200.00	Sports
2	Cricket World Cup Final	2025-08-20	14:30:00	50000	48000	2	1500.00	Sports

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

```
131
132     -- events name not start with 'x', 'y', 'z'
133 •   select * from event where event_name not like 'x%' and  event_name not like '%y' and event_name not like  '%z';
```

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

event_id	event_name	event_date	event_time	total_seats	available_seats	venue_id	ticket_price	event_type
1	The Dark Knight Rises	2025-07-15	19:00:00	300	250	1	200.00	Movie
2	Unbooked Test Movie	2025-07-30	17:00:00	100	100	1	250.00	Movie
3	Cricket World Cup Final	2025-08-20	14:30:00	50000	48000	2	1500.00	Sports
4	Rock Legends Concert	2025-09-10	20:00:00	5000	4500	3	2500.00	Concert
5	Comedy Night Live	2025-07-25	21:00:00	150	100	4	150.00	Movie
6	Annual Sports Meet	2025-08-01	09:00:00	10000	9000	5	500.00	Sports
7	Classical Music Gala	2025-09-22	18:00:00	800	750	3	1800.00	Concert
8	Summer Movie Marathon	2025-07-20	10:00:00	400	350	1	100.00	Movie
9	Football Championship	2025-10-05	16:00:00	30000	28000	2	1200.00	Sports
10	Jazz Fusion Night	2025-10-15	20:30:00	600	550	6	2000.00	Concert
11	AI fusion	2025-06-20	10:00:00	100	40	1	500.00	Concert
12	Kids Movie Festival	2025-07-18	11:00:00	200	180	7	80.00	Movie

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

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

```
128
129     -- query to List Events and Their Average Ticket Prices
130 •   select event_name , avg(ticket_price) as Average_Ticket_Price
131     from event group by event_name ;
```

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

event_name	Average_Ticket_Price
The Dark Knight Rises	200.000000
Cricket World Cup Final	1500.000000
Rock Legends Concert	2500.000000
Comedy Night Live	150.000000
Annual Sports Meet	500.000000
Classical Music Gala	1800.000000
Summer Movie Marathon	100.000000
Football Championship	1200.000000
Jazz Fusion Night	2000.000000
Kids Movie Festival	80.000000

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

```
132
133     -- Calculate the Total Revenue Generated by Events
134 •   select E.event_name , sum(B.total_cost) as Total_Revenue
135     from event E
136     join Booking B on E.event_id = B.event_id
137     group by event_name;
```

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

event_name	Total_Revenue
The Dark Knight Rises	1200.00
Cricket World Cup Final	7500.00
Rock Legends Concert	2500.00
Comedy Night Live	150.00
Annual Sports Meet	3000.00
Classical Music Gala	3600.00
Summer Movie Marathon	400.00
Football Championship	6000.00
Jazz Fusion Night	6000.00
Kids Movie Festival	160.00

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

```
138
139     -- find the event with the highest ticket sales
140 •   select E.event_name, sum(B.num_tickets) as TotalTicketsSold
141     from Event E
142     join Booking B on E.event_id = B.event_id
143     group by E.event_name
144     order by TotalTicketsSold desc
145     LIMIT 1;
146
```

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

event_name	TotalTicketsSold
The Dark Knight Rises	6

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

```
154     -- Calculate the Total Number of Tickets Sold for Each Event
155     select E.event_name,
156            coalesce((
157              select count(*)
158              from Booking B
159              where B.event_id = E.event_id
160            ), 0) as TicketsSold
161     from Event E;
```

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

event_name	TicketsSold
The Dark Knight Rises	3
Unbooked Test Movie	1
Cricket World Cup Final	1
Rock Legends Concert	1
Comedy Night Live	1
Annual Sports Meet	1
Classical Music Gala	1
Summer Movie Marathon	1
Football Championship	1
Jazz Fusion Night	1
AI fusion	0
Kids Movie Festival	0

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

```
159
160     -- Events with No Ticket Sales
161     select E.event_name-- , sum(B.num_tickets) as TotalTicketsSold
162     from Event E
163     left join Booking B on E.event_id = B.event_id
164     where booking_id is null;
```

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

event_name
AI fusion
Kids Movie Festival

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

```
159
160     -- User Who Has Booked the Most Tickets
161     select C.customer_name , sum(B.num_tickets) as TicketsBooked
162     from Customer C
163     left join Booking B on C.customer_id = B.booking_id
164     group by C.customer_name
165     order by TicketsBooked desc
166     limit 1;
167
```

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

customer_name	TicketsBooked
Frank Green	6

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

```
167
168     -- Events and the total number of tickets sold for each month
169     select E.event_name,date_format(B.booking_date, '%Y-%m') as BookingMonth , sum(B.num_tickets) as TicketSold
170     from Event E
171     join Booking B on E.event_id = B.booking_id
172     group by E.event_name , BookingMonth
173     order by E.event_name , BookingMonth;
174
```

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

event_name	BookingMonth	TicketSold
Annual Sports Meet	2025-06	1
Classical Music Gala	2025-06	6
Comedy Night Live	2025-06	3
Cricket World Cup Final	2025-06	5
Football Championship	2025-06	4
Jazz Fusion Night	2025-06	5
Kids Movie Festival	2025-06	3
Rock Legends Concert	2025-06	1
Summer Movie Marathon	2025-06	2
The Dark Knight Rises	2025-06	2

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

```
174
175     -- calculate the average Ticket Price for Events in Each Venue
176 •   select V.venue_name, E.event_name ,avg(E.ticket_price) as AverageTicketPrice
177     from Event E
178     join Venue V on E.event_id= V.venue_id
179     group by V.venue_name,E.event_name;
180
```

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

venue_name	event_name	AverageTicketPrice
Grand Cinema	The Dark Knight Rises	200.000000
City Stadium	Cricket World Cup Final	1500.000000
Concert Hall	Rock Legends Concert	2500.000000
Exhibition Center	Comedy Night Live	150.000000
Community Theater	Annual Sports Meet	500.000000
Museum of Art	Classical Music Gala	1800.000000
Science Center	Summer Movie Marathon	100.000000
Botanical Gardens	Football Championship	1200.000000
Waterfront Park	Jazz Fusion Night	2000.000000
The Jazz Club	Kids Movie Festival	80.000000

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

```
180
181     -- calculate the total Number of Tickets Sold for Each Event Type
182 •   select E.event_type , sum(B.num_tickets) as TotalTicketsSold
183     from Event E
184     left join Booking B on E.event_id = B.event_id
185     group by E.event_type;
186
```

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

event_type	TotalTicketsSold
Movie	13
Sports	16
Concert	6

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

```
186
187     -- calculate the total Revenue Generated by Events in Each Year
188 •   select  year(booking_date) as BookingYear,sum(total_cost) as TotalRevenue
189     from Booking
190     group by BookingYear
191     order by BookingYear;
192
```

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

BookingYear	TotalRevenue
2025	30510.00

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

```
192
193     -- users who have booked tickets for multiple events
194 •   select C.customer_name from Customer C
195     join Booking B ON C.customer_id = B.customer_id
196     GROUP BY C.customer_id, C.customer_name
197     HAVING COUNT(DISTINCT B.event_id) > 1;
198
```

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

customer_name
Alice Smith
Bob Johnson

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

```
198
199     -- calculate the Total Revenue Generated by Events for Each User
200 •   select C.customer_name , sum(B.total_cost) as TotalRevenue
201     from Customer C
202     join Booking B on C.customer_id = B.customer_id
203     group by customer_name;
204
```

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

customer_name	TotalRevenue
Alice Smith	560.00
Bob Johnson	7700.00
Charlie Brown	2500.00
Diana Prince	600.00
Eve Adams	150.00
Frank Green	3000.00
Grace Hopper	3600.00
Harry Potter	400.00
Ivy Rose	6000.00
Jack Sparrow	6000.00

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

```
204
205     -- calculate the Average Ticket Price for Events in Each Category and Venue
206 •   select V.venue_name, E.event_type, avg(E.ticket_price) as AverageTicketPrice
207     from Venue V
208     join Event E on V.venue_id = E.venue_id
209     group by V.venue_name, E.event_type
210     order by V.venue_name, E.event_type;
211
```

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

venue_name	event_type	AverageTicketPrice
City Stadium	Sports	1350.000000
Community Theater	Sports	500.000000
Concert Hall	Concert	2150.000000
Exhibition Center	Movie	150.000000
Grand Cinema	Movie	150.000000
Museum of Art	Concert	2000.000000
Science Center	Movie	80.000000

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

```
211
212     -- list Users and the Total Number of Tickets They've Purchased in the Last 30 Days
213 •   select C.customer_name , sum(B.num_tickets) as TotalTicketsPurchased
214     from Customer C
215     join Booking B on C.customer_id = B.customer_id
216     where B.booking_date >= date_sub(curdate(), interval 30 day)
217     group by C.customer_name
218     order by TotalTicketsPurchased desc;
219
```

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

customer_name	TotalTicketsPurchased
Bob Johnson	6
Frank Green	6
Ivy Rose	5
Alice Smith	4
Harry Potter	4
Diana Prince	3
Jack Sparrow	3
Grace Hopper	2
Charlie Brown	1
Eve Adams	1

Tasks 4: Subquery and its types:

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

```
223     -- TASK 4
224     -- Average Ticket Price for Events in Each Venue Using a Subquery
225 •   select V.venue_name,
226             coalesce((
227              select avg(E.ticket_price)
228               from Event E
229               where E.venue_id = V.venue_id
230             ), 0) as AverageTicketPrice
231     from Venue V;
```

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

venue_name	AverageTicketPrice
Grand Cinema	150.000000
City Stadium	1350.000000
Concert Hall	2150.000000
Exhibition Center	150.000000
Community Theater	500.000000
Museum of Art	2000.000000
Science Center	80.000000
Botanical Gardens	0.000000
Waterfront Park	0.000000
The Jazz Club	0.000000
Children's Museum ...	0.000000
Innovation Hub	0.000000

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

```
234     -- Events with More Than 50% of Tickets Sold using a subquery
235 •   select event_name, total_seats, available_seats
236     from Event
237     where (total_seats - available_seats) > (total_seats * 0.5);
```

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

event_name	total_seats	available_seats
AI fusion	100	40

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

```
243
244     -- Total Number of Tickets Sold for Each Event
245 •   select E.event_name,
246             coalesce((
247                 select sum(B.num_tickets)
248                 from booking B
249                 where B.event_id = E.event_id
250             ), 0) as Total_Tickets_Sold
251     from Event E ;
```

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

event_name	Total_Tickets_Sold
The Dark Knight Rises	6
Unbooked Test Movie	5
Cricket World Cup Final	1
Rock Legends Concert	1
Comedy Night Live	6
Annual Sports Meet	2
Classical Music Gala	4
Summer Movie Marathon	5
Football Championship	3
Jazz Fusion Night	2
AI fusion	0
Kids Movie Festival	0

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

```
245     -- Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery
246     select Customer_name from Customer C
247     where not exists
248      (select  1
249      from Booking B
250      where B.customer_id = C.customer_id
251     );
252
```

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

Customer_name
Jack Sparrow

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

```
254
255     -- Events with No Ticket Sales Using a NOT IN Subquery
256     select event_name
257     from Event
258     where event_id not in (select distinct event_id from Booking);
259
```

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

event_name
AI fusion
Kids Movie Festival

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

```
250
251     -- Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause
252     select event_type, sum(num_tickets_sold) AS TotalTicketsSold
253     from (
254         select E.event_type, B.num_tickets as num_tickets_sold
255         from Event E
256         join Booking B on E.event_id = B.event_id
257     ) as EventSales
258     group by event_type;
259
```

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

event_type	TotalTicketsSold
Movie	13
Sports	16
Concert	6

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

```
259
260     -- Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause
261     select event_name, ticket_price
262     from Event
263     where ticket_price > (select avg(ticket_price) from Event);
264
```

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

event_name	ticket_price
Cricket World Cup Final	1500.00
Rock Legends Concert	2500.00
Classical Music Gala	1800.00
Football Championship	1200.00
Jazz Fusion Night	2000.00

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

```
264
265     -- Total Revenue Generated by Events for Each User Using a Correlated Subquery
266     select
267         C.customer_name,
268         (select sum(B.total_cost) from Booking B where B.customer_id = C.customer_id) AS TotalRevenue
269     from Customer C;
```

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

customer_name	TotalRevenue
Alice Smith	560.00
Bob Johnson	7700.00
Charlie Brown	2500.00
Diana Prince	600.00
Eve Adams	150.00
Frank Green	3000.00
Grace Hopper	3600.00
Harry Potter	400.00
Ivy Rose	6000.00
Jack Sparrow	6000.00

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

```
270
271     -- Users Who Have Booked Tickets for Events (for example 'Grand Cinema') in a Given Venue Using a Subquery in the WHERE Clause
272     select distinct C.customer_name
273     from Customer C
274     where C.customer_id in (
275         select B.customer_id
276         from Booking B
277         join Event E on B.event_id = E.event_id
278         join Venue V on E.venue_id = V.venue_id
279         where V.venue_name = 'Grand Cinema'
280     );
```

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

customer_name
Alice Smith
Bob Johnson
Diana Prince
Harry Potter

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

```
281
282     -- Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY
283     select event_type, sum(tickets_sold) as TotalTicketsSold
284     from (
285         select E.event_type, B.num_tickets as tickets_sold
286         from Event E
287         join Booking B on E.event_id = B.event_id
288     ) as EventCategorySales
289     group by event_type;
```

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

event_type	TotalTicketsSold
Movie	13
Sports	16
Concert	6

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

```
291    -- Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT
292    select distinct C.customer_name, date_format(B.booking_date, '%Y-%m') as BookingMonth
293    from Customer C
294    join Booking B on C.customer_id = B.customer_id
295    order by C.customer_name, BookingMonth;
296
```

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

customer_name	BookingMonth
Alice Smith	2025-06
Bob Johnson	2025-06
Charlie Brown	2025-06
Diana Prince	2025-06
Eve Adams	2025-06
Frank Green	2025-06
Grace Hopper	2025-06
Harry Potter	2025-06
Ivy Rose	2025-06
Jack Sparrow	2025-06

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

```
304
305    -- Average Ticket Price for Events in Each Venue Using a Subquery
306    select V.venue_name, (
307        select avg(E.ticket_price)
308         from Event E
309         where E.venue_id = V.venue_id
310    ) as AverageTicketPrice
311    from Venue V
312    where V.venue_id in (select distinct venue_id from Event);
```

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

venue_name	AverageTicketPrice
Grand Cinema	262.500000
City Stadium	1350.000000
Concert Hall	2150.000000
Exhibition Center	150.000000
Community Theater	500.000000
Museum of Art	2000.000000
Science Center	80.000000