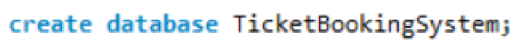
Assignment – 5

Ticket Booking System

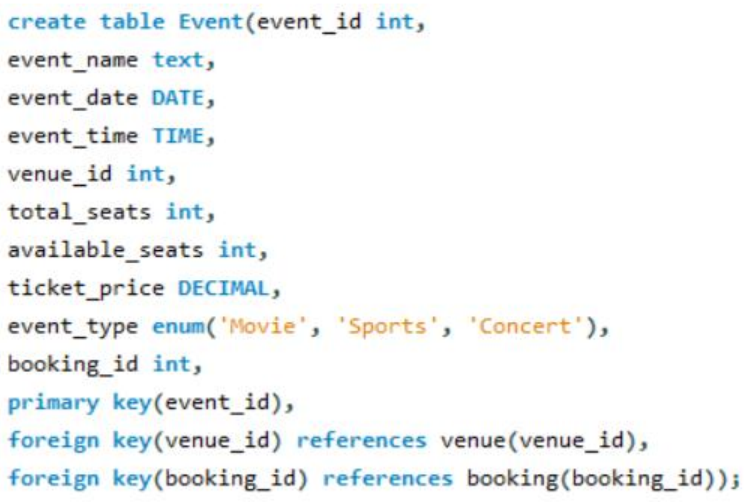
Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem".

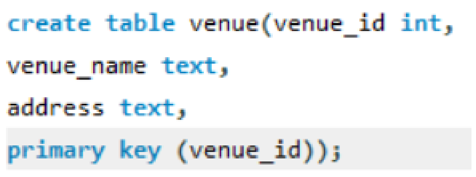


```
create database TicketBookingSystem;
```

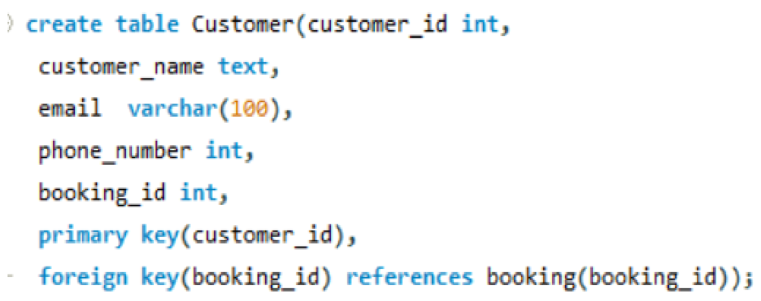
2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. • Venue • Event • Customers • Booking .



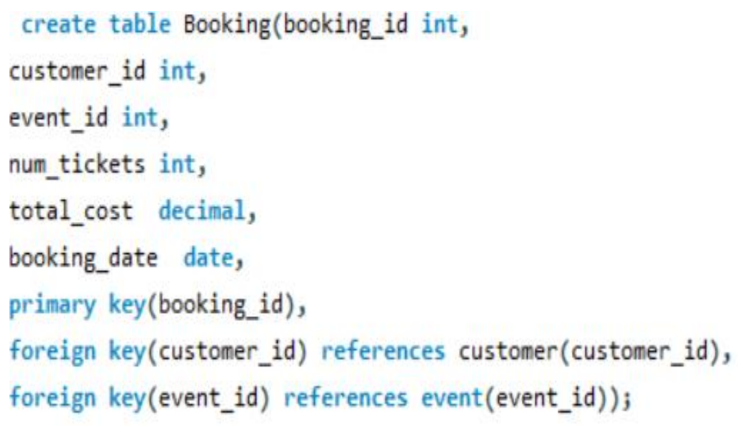
```
create table Event(event_id int,
event_name text,
event_date DATE,
event_time TIME,
venue_id int,
total_seats int,
available_seats int,
ticket_price DECIMAL,
event_type enum('Movie', 'Sports', 'Concert'),
booking_id int,
primary key(event_id),
foreign key(venue_id) references venue(venue_id),
foreign key(booking_id) references booking(booking_id));
```



```
create table venue(venue_id int,
venue_name text,
address text,
primary key (venue_id));
```

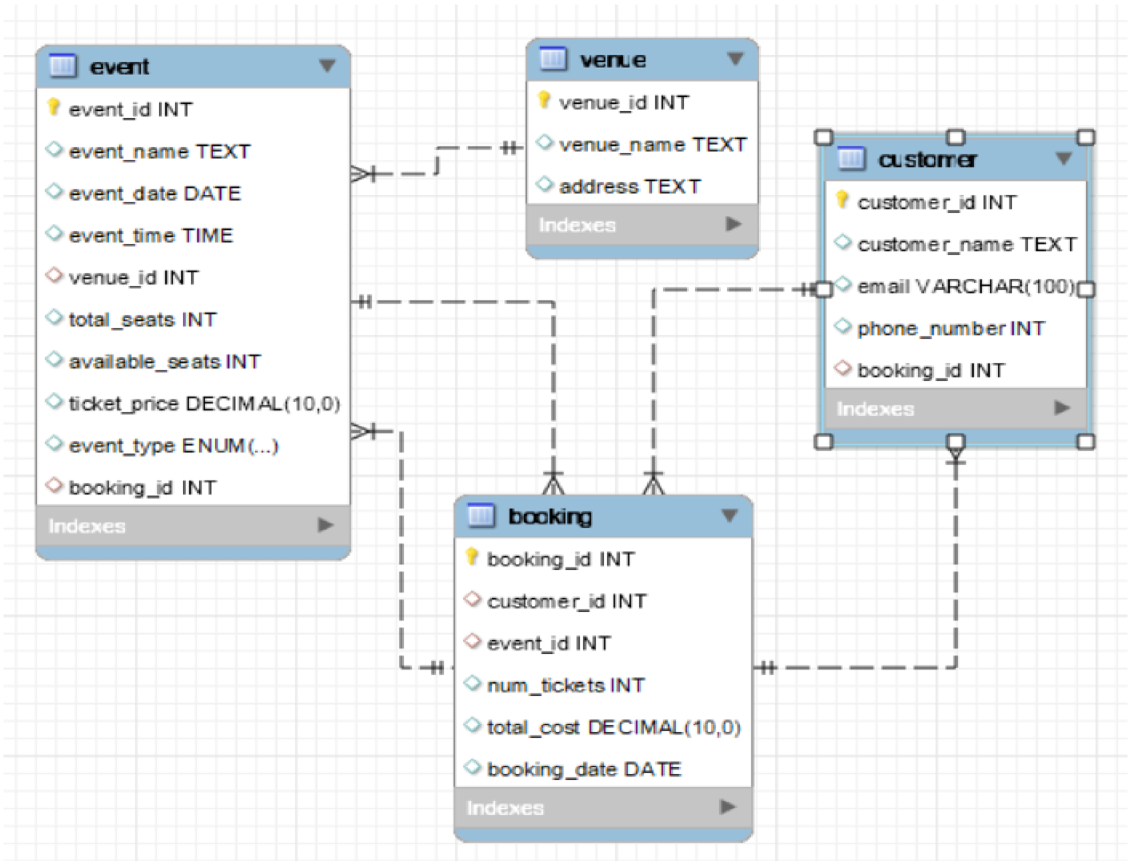


```
create table Customer(customer_id int,
  customer_name text,
  email  varchar(100),
  phone_number int,
  booking_id int,
  primary key(customer_id),
  foreign key(booking_id) references booking(booking_id));
```



```
create table Booking(booking_id int,
customer_id int,
event_id int,
num_tickets int,
total_cost  decimal,
booking_date  date,
primary key(booking_id),
foreign key(customer_id) references customer(customer_id),
foreign key(event_id) references event(event_id));
```

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



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, 'Convention Center', '123 Main Street'),
(2, 'Stadium Arena', '456 Park Avenue'),
(3, 'City Hall Auditorium', '789 Broadway'),
(4, 'Grand Movieplex', '321 Oak Street'),
(5, 'Sports Stadium', '567 Pine Avenue'),
(6, 'Harmony Concert Hall', '890 Elm Street'),
(7, 'Film City Theater', '234 Maple Avenue'),
(8, 'Outdoor Arena', '876 Cedar Street'),
(9, 'Central Theater', '543 Birch Avenue'),
(10, 'Music Palace', '987 Redwood Street');
```

```
insert into event(event_id, event_name, event_date, event_time, venue_id, total_seats, available_seats,ticket_price,event_type,booking_id)
 values (1, 'Movie Night', '2024-01-14', '18:30:00', 1, 100, 80, 10.00, 'Movie',1),
(2, 'Football Championship', '2024-02-05', '15:00:00', 2, 500, 300, 25.00, 'Sports',2),
(3, 'Concert Live', '2024-03-20', '20:00:00', 3, 200, 150, 40.00, 'Concert',3),
(4, 'Basketball Game', '2024-04-10', '19:15:00', 4, 300, 200, 20.00, 'Sports',4),
(5, 'Theater Play', '2024-05-02', '19:30:00', 5, 150, 120, 15.00, 'Concert',5),
(6, 'Rock Concert', '2024-06-15', '21:00:00', 6, 250, 180, 35.00, 'Concert',6),
(7, 'Soccer Match', '2024-07-08', '17:45:00', 7, 400, 250, 30.00, 'Sports',7),
(8, 'Movie Premiere', '2024-08-19', '19:00:00', 8, 120, 90, 12.00, 'Movie',8),
(9, 'Ice Hockey Game', '2024-09-03', '18:45:00', 9, 350, 220, 28.00, 'Sports',9),
(10, 'Jazz Festival', '2024-10-12', '20:30:00', 10, 180, 140, 18.00, 'Concert',10);
```

```
insert into booking(booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) values
(1,1,1, 2, 20.00, '2024-01-14'),
(2,2, 2, 4, 100.00, '2024-02-05'),
(3,3,3, 3, 120.00, '2024-03-20'),
(4,4, 4, 2, 40.00, '2024-04-10'),
(5,5, 5, 1, 15.00, '2024-05-02'),
(6,6, 6, 5, 175.00, '2024-06-15'),
(7,7, 7, 3, 90.00, '2024-07-08'),
(8,8, 8, 2, 24.00, '2024-08-19'),
(9,9, 9, 4, 60.00, '2024-09-03'),
(10,10,10, 3, 54.00, '2024-10-12');
```

```
insert into customer(customer_id, customer_name, email, phone_number, booking_id) values
(1, 'Alice Johnson', 'alice@email.com', 55512378, 1),
(2, 'Bob Miller', 'bob@email.com', 55598765, 2),
(3, 'Chris Robinson', 'chris@email.com', 55555556, 3),
(4, 'Daniel Wright', 'daniel@email.com', 55555523, 4),
(5, 'Eva Jones', 'eva@email.com', 55555512, 5),
(6, 'Jane Smith', 'jane@email.com', 98765432, 6),
(7, 'Matthew Taylor', 'matthew@email.com', 55555534, 7),
(8, 'Olivia Clark', 'olivia@email.com', 55555578, 8),
(9, 'Sophia Davis', 'sophia@email.com', 55555590, 9),
(10, 'John Doe', 'john@email.com', 12345678, 10);
```

2. Write a SQL query to list all Events.

```
select event_name from event;
```

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

```
select * from event where available_seats > 0;
```

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

```
select event_name from event where event_name like "%cup%";
```

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

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

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

```
select event_name,event_date from event where event_date between "2024-01-01" and "2024-08-01";
```

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

```
select event_name,available_seats from event where event_name like"%concert%";
```

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

```
select customer_name from customer limit 5 offset 5;
```

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

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

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

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

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

```
select * from event where total_seats > 15000;
```

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

```
select event_name from event where event_name not like "x%" and event_name not like "y%" and event_name not like "z%";
```

<u>Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:</u>

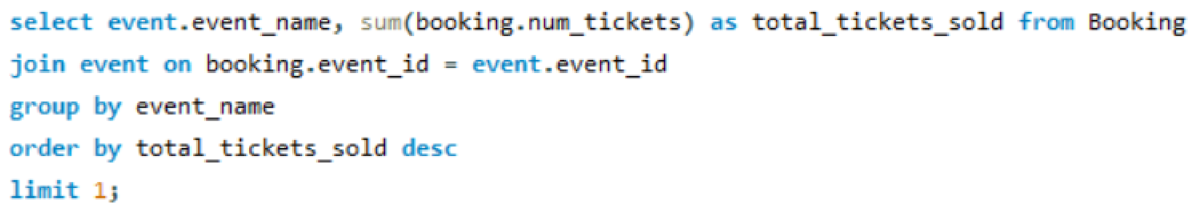
1. Write a SQL query to List Events and Their Average Ticket Prices.

```
select event_name,avg(ticket_price) as averageTicketPrice from event
group by event_name;
```

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

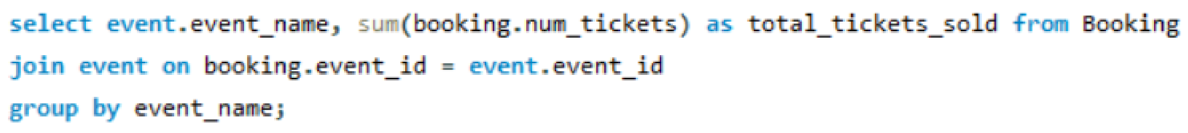
```
select sum(total_cost) as TotalRevenue from booking;
```

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



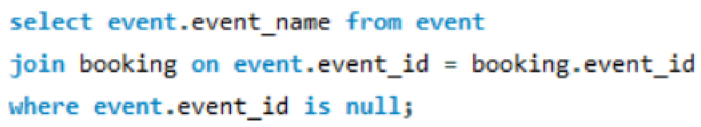
```
select event.event_name, sum(booking.num_tickets) as total_tickets_sold from Booking
join event on booking.event_id = event.event_id
group by event_name
order by total_tickets_sold desc
limit 1;
```

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



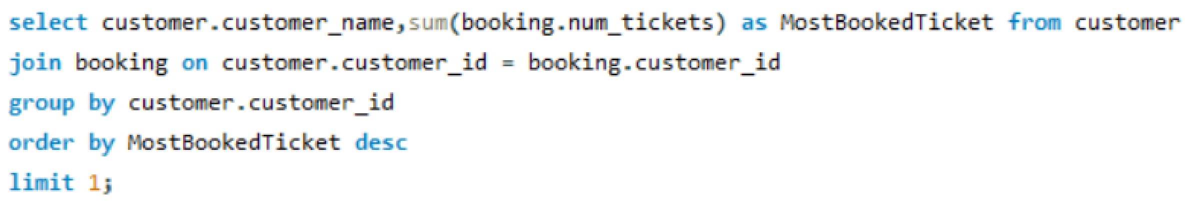
```
select event.event_name, sum(booking.num_tickets) as total_tickets_sold from Booking
join event on booking.event_id = event.event_id
group by event_name;
```

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



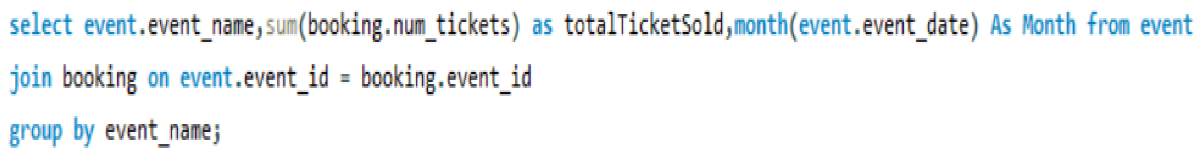
```
select event.event_name from event
join booking on event.event_id = booking.event_id
where event.event_id is null;
```

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



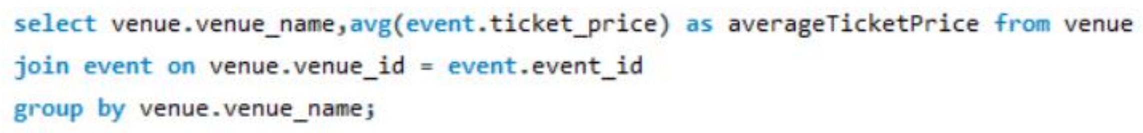
```
select customer.customer_name,sum(booking.num_tickets) as MostBookedTicket from customer
join booking on customer.customer_id = booking.customer_id
group by customer.customer_id
order by MostBookedTicket desc
limit 1;
```

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



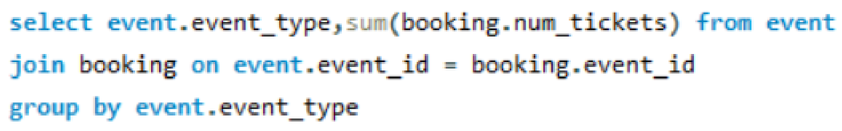
```
select event.event_name,sum(booking.num_tickets) as totalTicketSold,month(event.event_date) As Month from event
join booking on event.event_id = booking.event_id
group by event_name;
```

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



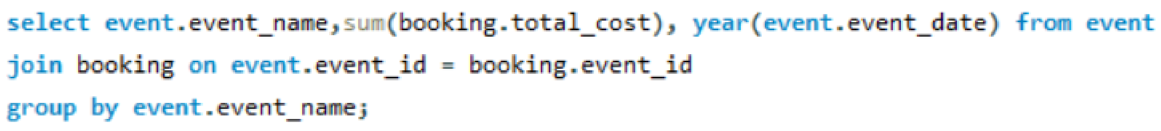
```
select venue.venue_name,avg(event.ticket_price) as averageTicketPrice from venue
join event on venue.venue_id = event.event_id
group by venue.venue_name;
```

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



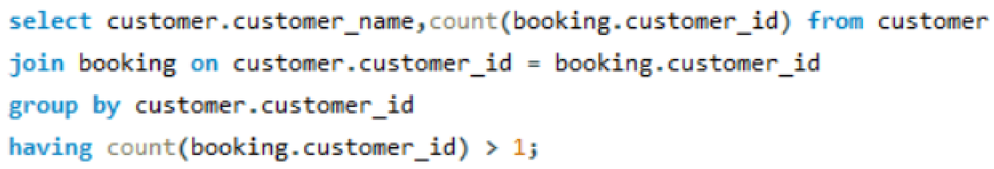
```
select event.event_type,sum(booking.num_tickets) from event
join booking on event.event_id = booking.event_id
group by event.event_type
```

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



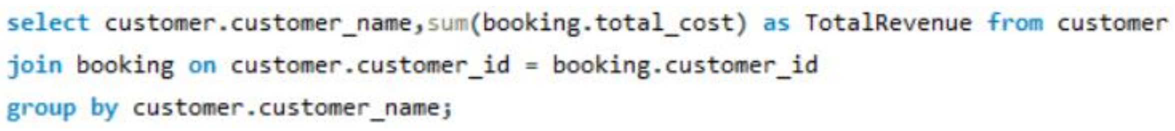
```
select event.event_name,sum(booking.total_cost), year(event.event_date) from event
join booking on event.event_id = booking.event_id
group by event.event_name;
```

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



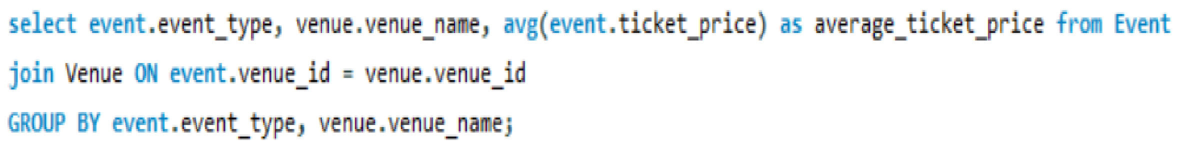
```
select customer.customer_name,count(booking.customer_id) from customer
join booking on customer.customer_id = booking.customer_id
group by customer.customer_id
having count(booking.customer_id) > 1;
```

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



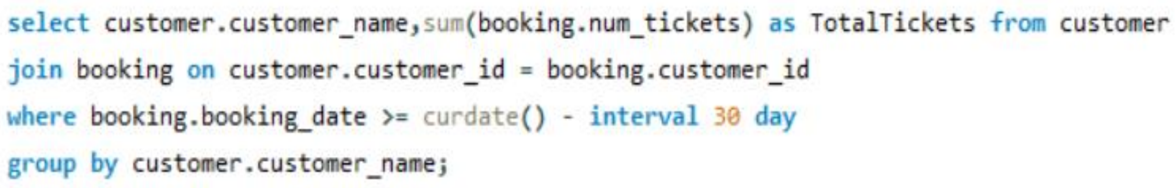
```
select customer.customer_name,sum(booking.total_cost) as TotalRevenue from customer
join booking on customer.customer_id = booking.customer_id
group by customer.customer_name;
```

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



```
select event.event_type, venue.venue_name, avg(event.ticket_price) as average_ticket_price from Event
join Venue ON event.venue_id = venue.venue_id
GROUP BY event.event_type, venue.venue_name;
```

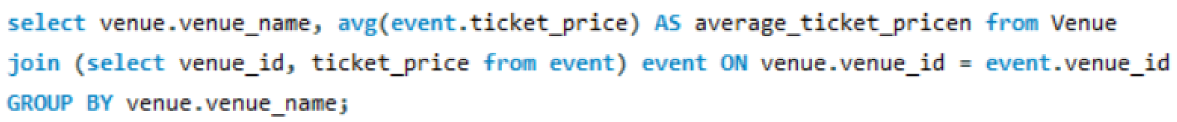
14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.



```
select customer.customer_name,sum(booking.num_tickets) as TotalTickets from customer
join booking on customer.customer_id = booking.customer_id
where booking.booking_date >= curdate() - interval 30 day
group by customer.customer_name;
```

Tasks 4: Subquery and its types

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



```
select venue.venue_name, avg(event.ticket_price) AS average_ticket_pricen from Venue
join (select venue_id, ticket_price from event) event ON venue.venue_id = event.venue_id
GROUP BY venue.venue_name;
```

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) );
```

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

```sql
select event_name, (
    select SUM(num_tickets)
    from Booking
    WHERE event.event_id = booking.event_id
) as total_tickets_sold from Event;
```

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

```sql
select customer_name from customer where not exists
(select booking_id from booking
where customer.customer_id = booking.booking_id);
```

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

```sql
select event_name from event where event_id not in
(select distinct event_id from booking );
```

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

```sql
select event.event_type,sum(booking.num_tickets) from event
join booking on event.event_id = booking.event_id
group by event.event_type
```

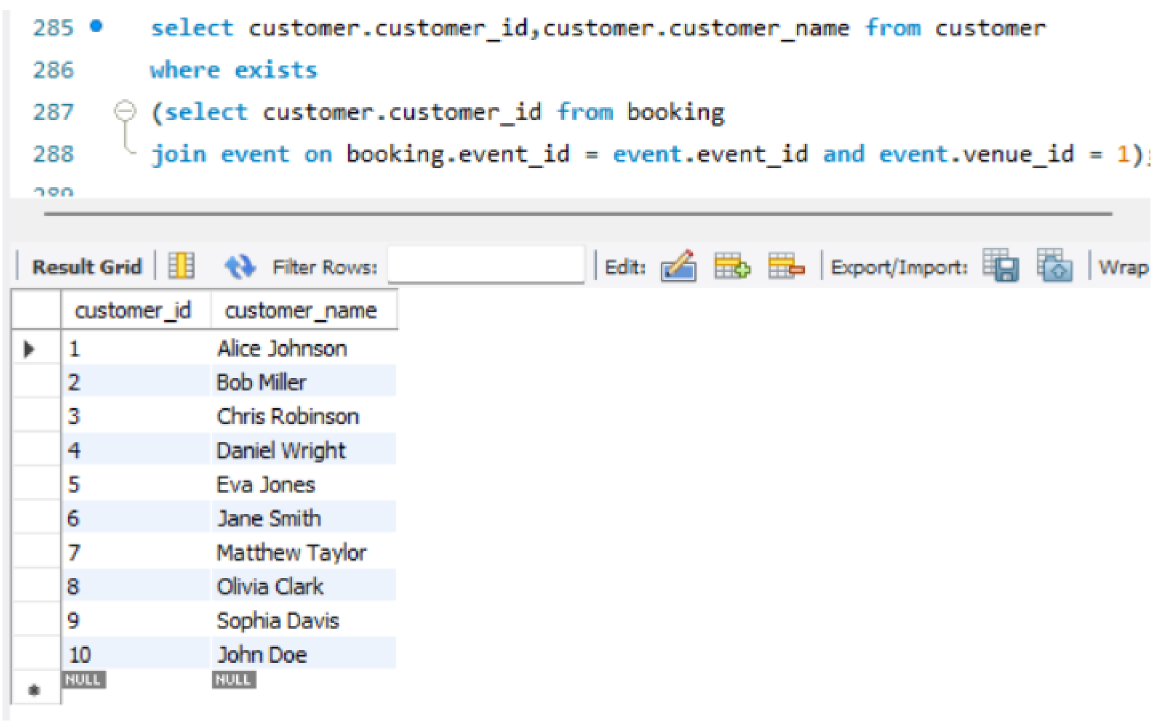
7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.

```sql
select event_name, ticket_price
from event where ticket_price > (
    select avg(ticket_price)
   from event );
```

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

```
select  customer_name, (
    select SUM(total_cost)
    FROM Booking
    WHERE customer.customer_id = booking.customer_id
) as total_revenue
from Customer;
```

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



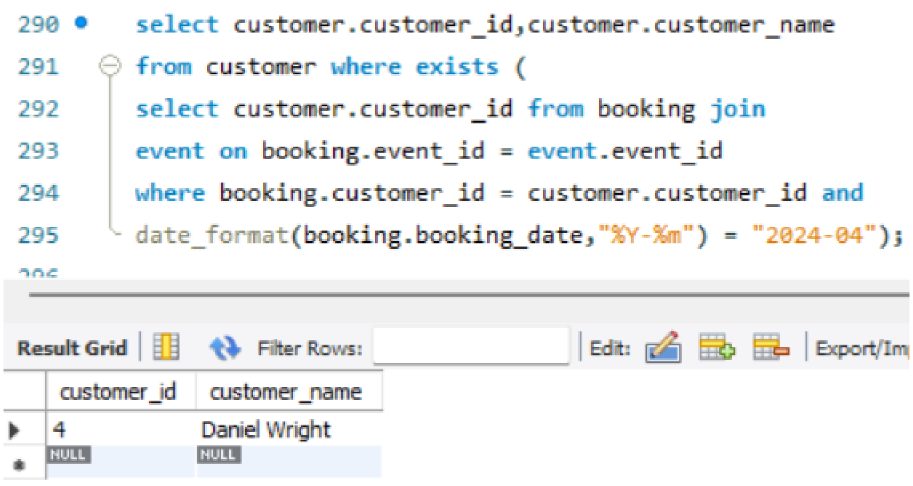
```
select customer.customer_id,customer.customer_name from customer
where exists
(select customer.customer_id from booking
join event on booking.event_id = event.event_id and event.venue_id = 1);
```

customer_id	customer_name
1	Alice Johnson
2	Bob Miller
3	Chris Robinson
4	Daniel Wright
5	Eva Jones
6	Jane Smith
7	Matthew Taylor
8	Olivia Clark
9	Sophia Davis
10	John Doe
NULL	NULL

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

```
select event_type, (
    select SUM(num_tickets)
    from Booking
    where event.event_id = booking.event_id
) as total_tickets_sold
from event
group by event.event_type;
```

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

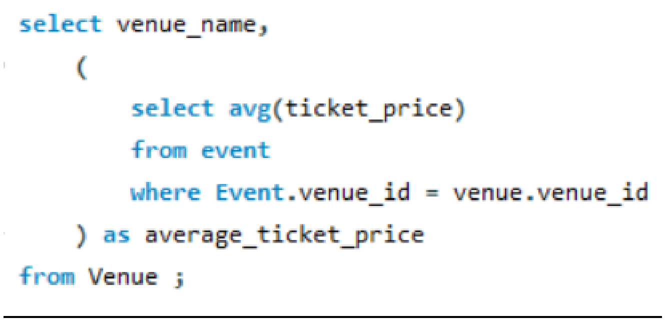


```
select customer.customer_id,customer.customer_name
from customer where exists (
select customer.customer_id from booking join
event on booking.event_id = event.event_id
where booking.customer_id = customer.customer_id and
date_format(booking.booking_date,"%Y-%m") = "2024-04");
```

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

customer_id	customer_name
4	Daniel Wright
NULL	NULL

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



```
select venue_name,
    (
        select avg(ticket_price)
        from event
        where Event.venue_id = venue.venue_id
    ) as average_ticket_price
from Venue ;
```