Assignment 5 (TicketBookingSystem)

- ***Task : 1 Database Design***
  - *Create the database named "TicketBookingSystem"*
    - ```
      create database TicketBookingSystem;
      use TicketBookingSystem;
      ```
  - *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(100),
        address varchar(100)
        );
        ```
    - *EventTable*
      - ```
        create table EventTable(
        booking_id int,
        event_id int primary key,
        event_name varchar(100),
        event_date varchar(100),
        event_time time,
        venue_id int,
        total_seats varchar (100),
        available_seats varchar (100),
        ticket_price decimal(10,2),
        event_type ENUM ('Movie', 'Sports', 'Concert'),
        foreign key (booking_id) references BookingTable(booking_id),
        foreign key (venue_id) references venue(venue_id)
        );
        ```
    - *CustomerTable*
      - ```
        create table CustomerTable(
        booking_id int,
        customer_id  int primary key,
        customer_name varchar(50),
        email varchar(50),
        phone_number varchar(50),
        ```

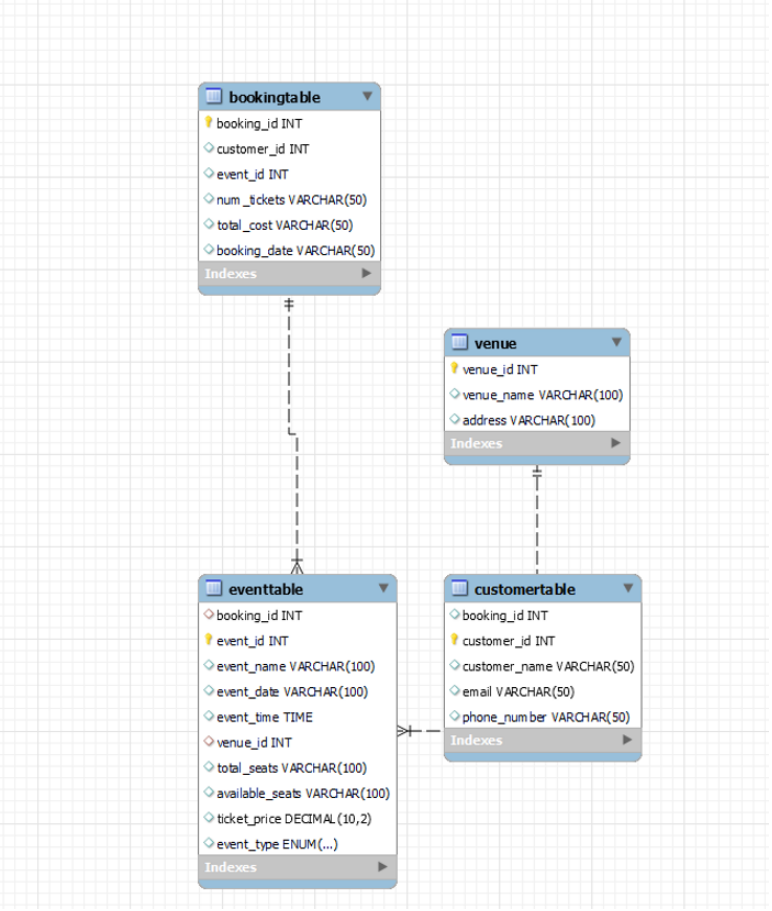
```
foreign key (booking_id) references BookingTable(booking_id)
);
```

- *BookingTable*
  - ```
    create table BookingTable(
    booking_id int primary key,
    customer_id int,
    event_id int,
    num_tickets varchar(50),
    total_cost varchar(50),
    booking_date varchar(50),
    ```

    ```
    foreign key (customer_id) references CustomerTable(customer_id),
    foreign key (event_id) references EventTable(event_id)
    );
    ```

- *Create an ERD (Entity Relationship Diagram) for the database.*
- *4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.*

- ***Code Snippet : Task 1***
- ***ERD (Entity Relationship Diagram)***


    ```

- Ta***sks 2: Select, Where, Between, AND, LIKE***

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

   a. *Code snippet*

      i. *Venue*

      1.
```
insert into venue( venue_id , venue_name, address)
values
 ('1', 'Venue1', 'Address'),
('2', 'Venue2', 'Address2'),
('3', 'Venue3', 'Address3'),
('4', 'Venue4', 'Address4'),
('5', 'Venue5', 'Address5'),
('6', 'Venue6', 'Address6'),
('7', 'Venue7', 'Address7'),
('8', 'Venue8', 'Address8'),
('9', 'Venue9', 'Address9'),
('10', 'Venue10', 'Address10');
```

venue_id	venue_name	address
1	Venue1	Address
2	Venue2	Address2
3	Venue3	Address3
4	Venue4	Address4
5	Venue5	Address5
6	Venue6	Address6
7	Venue7	Address7

      ii. *EventTable*

```
insert into EventTable (event_id, event_name, event_date, event_time,
venue_id, total_seats, available_seats, ticket_price, event_type,
booking_id)
values
(1, 'Event1', '2024-01-20', '12:30:00', 1, '100', '100', 50.00,
'Concert', 1),
(2, 'Event2', '2024-02-15', '15:45:00', 2, '120', '120', 45.00, 'Sports',
2),
(3, 'Event3', '2024-03-10', '18:00:00', 3, '150', '150', 60.00, 'Movie',
3),
(4, 'Event4', '2024-04-05', '20:30:00', 4, '80', '80', 40.00, 'Concert',
4),
(5, 'Event5', '2024-05-22', '14:00:00', 5, '200', '200', 55.00, 'Sports',
5),
(6, 'Event6', '2024-06-18', '17:15:00', 6, '110', '110', 35.00, 'Movie',
6),
(7, 'Event7', '2024-07-12', '19:45:00', 7, '130', '130', 75.00,
'Concert', 7),
(8, 'Event8', '2024-08-28', '16:00:00', 8, '90', '90', 48.00, 'Sports',
8),
(9, 'Event9', '2024-09-14', '21:00:00', 9, '180', '180', 42.00, 'Movie',
9),
```

```
(10, 'Event10', '2024-10-30', '13:30:00', 10, '160', '160', 65.00,
'Concert', 10);
```

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_
1	1	Event1	2024-01-20	12:30:00	1	100	100	50.00	Concert
2	2	Event2	2024-02-15	15:45:00	2	120	120	45.00	Sports
3	3	Event3	2024-03-10	18:00:00	3	150	150	60.00	Movie
4	4	Event4	2024-04-05	20:30:00	4	80	80	40.00	Concert
5	5	Event5	2024-05-22	14:00:00	5	200	200	55.00	Sports
6	6	Event6	2024-06-18	17:15:00	6	110	110	35.00	Movie
7	7	Event7	2024-07-12	19:45:00	7	130	130	75.00	Concert

iii. *CustomerTable*

```
insert into
CustomerTable(customer_id,customer_name,email,phone_number,booking_id)
values
 (1001, 'John Doe', 'john.doe@example.com', '+1234567890', 1),
(1002, 'Jane Smith', 'jane.smith@example.com', '+9876543210', 2),
(1003, 'Bob Johnson', 'bob.johnson@example.com', '+1122334455', 3),
(1004, 'Alice Lee', 'alice.lee@example.com', '+9988776655', 4),
(1005, 'Charlie Brown', 'charlie.brown@example.com', '+1122334455', 5),
(1006, 'Eva Martinez', 'eva.martinez@example.com', '+9988776655', 6),
(1007, 'David Wang', 'david.wang@example.com', '+1122334455', 7),
(1008, 'Sophia Kim', 'sophia.kim@example.com', '+9988776655', 8),
(1009, 'Michael Garcia', 'michael.garcia@example.com', '+1122334455', 9),
(1010, 'Olivia Rodriguez', 'olivia.rodriguez@example.com', '+9988776655',
10);
```

booking_id	customer_id	customer_name	email	phone_number
1	1001	John Doe	john.doe@example.com	+1234567890
2	1002	Jane Smith	jane.smith@example.com	+9876543210
3	1003	Bob Johnson	bob.johnson@example.com	+1122334455
4	1004	Alice Lee	alice.lee@example.com	+9988776655
5	1005	Charlie Brown	charlie.brown@example.com	+1122334455
6	1006	Eva Martinez	eva.martinez@example.com	+9988776655
7	1007	David Wang	david.wang@example.com	+1122334455

iv. *BookingTable*

```
insert into BookingTable(booking_id,
customer_id,event_id,total_cost,booking_date)
values
 (1, 1001, 1, 75.00, '2024-01-25 14:45:00'),
(2, 1002, 3, 50.00, '2024-02-10 18:30:00'),
(3, 1003, 5, 100.00, '2024-03-05 20:15:00'),
(4, 1004, 2, 60.00, '2024-04-12 16:00:00'),
(5, 1005, 4, 85.00, '2024-05-20 19:30:00'),
(6, 1006, 6, 70.00, '2024-06-08 15:45:00'),
(7, 1007, 8, 120.00, '2024-07-15 21:00:00'),
(8, 1008, 7, 90.00, '2024-08-28 17:30:00'),
(9, 1009, 9, 55.00, '2022-09-14 14:00:00'),
(10, 1010, 10, 110.00, '2023-10-30 19:45:00');
```

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1	1001	1	NULL	75.00	2024-01-25 14:45:00
2	1002	3	NULL	50.00	2024-02-10 18:30:00
3	1003	5	NULL	100.00	2024-03-05 20:15:00
4	1004	2	NULL	60.00	2024-04-12 16:00:00
5	1005	4	NULL	85.00	2024-05-20 19:30:00
6	1006	6	NULL	70.00	2024-06-08 15:45:00
7	1007	8	NULL	120.00	2024-07-15 21:00:00

2. *Write a SQL query to list all Events.*
   a. `select * from eventtable;`
3. *Write a SQL query to select events with available tickets.*
   a. `SELECT *FROM EventTable WHERE available_seats > 0;`

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_
1	1	Event1	2024-01-20	12:30:00	1	100	100	50.00	Concert
2	2	Event2	2024-02-15	15:45:00	2	120	120	45.00	Sports
3	3	Event3	2024-03-10	18:00:00	3	150	150	60.00	Movie
4	4	Event4	2024-04-05	20:30:00	4	80	80	40.00	Concert
5	5	Event5	2024-05-22	14:00:00	5	200	200	55.00	Sports

4. *Write a SQL query to select events name partial match with ‘cup’.*
   a. `SELECT *FROM EventTable WHERE event_name LIKE '%cup%';`

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_typ
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. *Write a SQL query to select events with ticket price range is between 1000 to 2500.*
   a. `SELECT *FROM EventTable WHERE ticket_price BETWEEN 1000 AND 2500;`

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_typ
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.*
   a.
```
SELECT *FROM EventTable
WHERE event_date BETWEEN '2024-02-01' AND '2024-03-01';
```

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_typ
2	2	Event2	2024-02-15	15:45:00	2	120	120	45.00	Sports
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.*

   a.
```
SELECT *FROM EventTable
WHERE available_seats > 0 AND event_name LIKE '%Concert%';
```

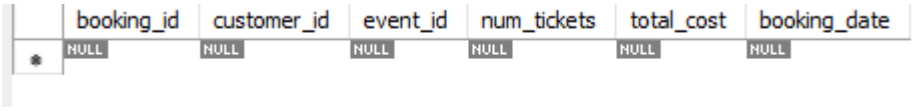
8. *Write a SQL query to retrieve users in batches of 5, starting from the 6th user.*

   a.
```
SELECT *FROM CustomerTable
ORDER BY customer_id
LIMIT 5 OFFSET 5;
```

booking_id	customer_id	customer_name	email	phone_number
6	1006	Eva Martinez	eva.martinez@example.com	+9988776655
7	1007	David Wang	david.wang@example.com	+1122334455
8	1008	Sophia Kim	sophia.kim@example.com	+9988776655
9	1009	Michael Garcia	michael.garcia@example.com	+1122334455
10	1010	Olivia Rodriguez	olivia.rodriguez@example.com	+9988776655
NULL	NULL	NULL	NULL	NULL

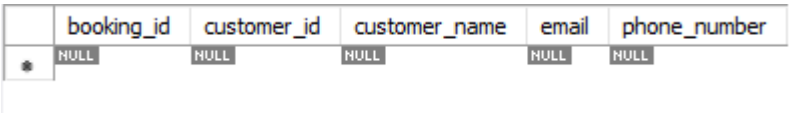
9. *Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.*

   a.
```
SELECT  *FROM BookingTable
WHERE CAST(num_tickets AS SIGNED) > 4;
```



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

    a.
```
SELECT *
FROM CustomerTable
WHERE phone_number LIKE '%000';
```



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

a.
```
SELECT *
FROM EventTable
WHERE CAST(total_seats AS SIGNED) > 15000
ORDER BY total_seats;
```

	booking_id	customer_id	customer_name	email	phone_number
*	NULL	NULL	NULL	NULL	NULL

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

a.
```
SELECT *
FROM EventTable
WHERE SUBSTRING(event_name, 1, 1) NOT IN ('x', 'y', 'z');
```

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_
1	1	Event1	2024-01-20	12:30:00	1	100	100	50.00	Concert
2	2	Event2	2024-02-15	15:45:00	2	120	120	45.00	Sports
3	3	Event3	2024-03-10	18:00:00	3	150	150	60.00	Movie
4	4	Event4	2024-04-05	20:30:00	4	80	80	40.00	Concert
5	5	Event5	2024-05-22	14:00:00	5	200	200	55.00	Sports

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

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

a.
```
SELECT event_id,event_name,
AVG(ticket_price) AS average_ticket_price
FROM EventTable
GROUP BY event_id, event_name;
```

event_id	event_name	average_ticket_price
1	Event1	50.000000
2	Event2	45.000000
3	Event3	60.000000
4	Event4	40.000000
5	Event5	55.000000
6	Event6	35.000000

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

a.
```
SELECTBookingTable.event_id,EventTable.event_name,
SUM(CAST(BookingTable.total_cost AS DECIMAL(10, 2))) AS total_revenue
FROM BookingTable
JOIN EventTable ON BookingTable.event_id = EventTable.event_id
GROUP BY
BookingTable.event_id, EventTable.event_name
LIMIT 0, 1000;
```

event_id	event_name	total_revenue
1	Event1	75.00
3	Event3	50.00
5	Event5	100.00
2	Event2	60.00
4	Event4	85.00
6	Event6	70.00

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

   a.
```
SELECT e.event_id,e.event_name,
SUM(CAST(b.num_tickets AS SIGNED)) AS total_tickets_sold
FROM BookingTable b
JOIN EventTable e ON b.event_id = e.event_id
GROUP BY e.event_id, e.event_name
ORDER BY total_tickets_sold DESC
LIMIT 1;
```

event_id	event_name	total_tickets_sold
1	Event1	NULL

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

   a.
```
SELECTevent_id,event_name,
SUM(CAST(num_tickets AS SIGNED)) AS total_tickets_sold
FROMcBookingTable
JOIN EventTable ON BookingTable.event_id = EventTable.event_id
GROUP BY event_id, event_name;
```

event_id	event_name	total_tickets_sold
1	Event1	NULL
3	Event3	NULL
5	Event5	NULL
2	Event2	NULL
4	Event4	NULL
6	Event6	NULL

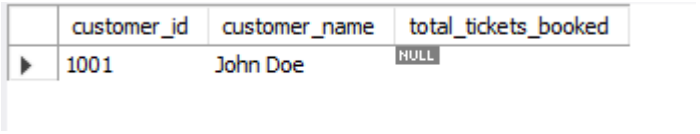
5. *Write a SQL query to Find Events with No Ticket Sales.*

   a.
```
SELECT *
FROM EventTable
WHERE event_id NOT IN (SELECT DISTINCT event_id FROM BookingTable);
```

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_typ
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

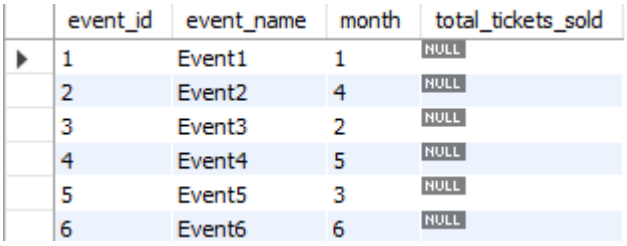
a.
```
SELECT CustomerTable.customer_id,CustomerTable.customer_name,
SUM(CAST(BookingTable.num_tickets AS SIGNED)) AS total_tickets_booked
FROM BookingTable
JOIN CustomerTable ON BookingTable.customer_id = CustomerTable.customer_id
GROUP BY
CustomerTable.customer_id, CustomerTable.customer_name
ORDER BY total_tickets_booked DESC
LIMIT 1;
```



customer_id	customer_name	total_tickets_booked
1001	John Doe	NULL

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

a.
```
SELECT EventTable.event_id,EventTable.event_name,
MONTH(BookingTable.booking_date) AS month,
SUM(CAST(BookingTable.num_tickets AS SIGNED)) AS total_tickets_sold
FROM BookingTable
JOIN EventTable ON BookingTable.event_id = EventTable.event_id
GROUP BY
EventTable.event_id, EventTable.event_name, month
ORDER BY
EventTable.event_id, month
LIMIT 0, 1000;
```



event_id	event_name	month	total_tickets_sold
1	Event1	1	NULL
2	Event2	4	NULL
3	Event3	2	NULL
4	Event4	5	NULL
5	Event5	3	NULL
6	Event6	6	NULL

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

a.
```
SELECT venue.venue_id,venue.venue_name,
AVG(EventTable.ticket_price) AS average_ticket_price
FROM EventTable
JOIN venue ON EventTable.venue_id = venue.venue_id
GROUP BY venue.venue_id, venue.venue_name
LIMIT 0, 1000;
```

venue_id	venue_name	average_ticket_price
1	Venue1	50.000000
2	Venue2	45.000000
3	Venue3	60.000000
4	Venue4	40.000000
5	Venue5	55.000000
6	Venue6	35.000000

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

   a.
```
SELECT event_type,
SUM(CAST(num_tickets AS SIGNED)) AS total_tickets_sold
FROM BookingTable
JOIN EventTable ON BookingTable.event_id = EventTable.event_id
GROUP BY
event_type;
```

event_type	total_tickets_sold
Concert	NULL
Movie	NULL
Sports	NULL

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

    a.
```
SELECT YEAR(booking_date) AS year,
SUM(CAST(total_cost AS DECIMAL(10, 2))) AS total_revenue
FROM BookingTable
GROUP BY year;
```

year	total_revenue
2024	815.00

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

    a.
```
SELECT customer_id, customer_name
FROM CustomerTable
WHERE
customer_id IN ( SELECT customer_id
FROM BookingTable
GROUP BY customer_id
HAVING
COUNT(DISTINCT event_id) > 1
);
```

customer_id	customer_name
NULL	NULL

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

    a.
```
SELECT CustomerTable.customer_id, CustomerTable.customer_name,
SUM(CAST(BookingTable.total_cost AS DECIMAL(10, 2))) AS total_revenue
FROM BookingTable
JOIN CustomerTable ON BookingTable.customer_id = CustomerTable.customer_id
GROUP BY
CustomerTable.customer_id, CustomerTable.customer_name
LIMIT 0, 1000;
```

customer_id	customer_name	total_revenue
1001	John Doe	75.00
1002	Jane Smith	50.00
1003	Bob Johnson	100.00
1004	Alice Lee	60.00
1005	Charlie Brown	85.00

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

    a.
```
SELECT event_type,venue_name,
AVG(ticket_price) AS average_ticket_price
FROM EventTable
JOIN venue ON EventTable.venue_id = venue.venue_id
GROUP BY event_type, venue_name;
```

event_type	venue_name	average_ticket_price
Concert	Venue1	50.000000
Sports	Venue2	45.000000
Movie	Venue3	60.000000
Concert	Venue4	40.000000
Sports	Venue5	55.000000

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

    a.
```
SELECT CustomerTable.customer_id,
CustomerTable.customer_name,
SUM(CAST(BookingTable.num_tickets AS SIGNED)) AS total_tickets_purchased
FROM BookingTable
JOIN CustomerTable ON BookingTable.customer_id = CustomerTable.customer_id
WHERE
booking_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
GROUP BY CustomerTable.customer_id, CustomerTable.customer_name
LIMIT 0, 1000;
```

	customer_id	customer_name	total_tickets_purchased
▶	1001	John Doe	NULL
	1002	Jane Smith	NULL
	1003	Bob Johnson	NULL
	1004	Alice Lee	NULL
	1005	Charlie Brown	NULL

TASK 4 : Subquery and its types

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

   a.
```
SELECT venue_id,venue_name,
(SELECT AVG(ticket_price) FROM EventTable WHERE venue_id = venue.venue_id) AS
average_ticket_price
FROM venue;
```

	venue_id	venue_name	average_ticket_price
▶	1	Venue1	50.000000
	2	Venue2	45.000000
	3	Venue3	60.000000
	4	Venue4	40.000000
	5	Venue5	55.000000

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

   a.
```
SELECT EventTable.*
FROM EventTable
JOIN (SELECT event_id, COUNT(*) AS booking_count
FROM BookingTable
GROUP BY event_id
) AS EventBookings ON EventTable.event_id = EventBookings.event_id
WHERE EventBookings.booking_count > 0.5 * EventTable.total_seats
LIMIT 0, 1000;
```

booking_id	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_typ

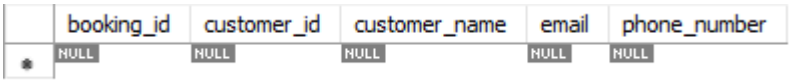
3. *Calculate the Total Number of Tickets Sold for Each Event.*

   a.
```
SELECT event_id,event_name,
(SELECT SUM(CAST(num_tickets AS SIGNED)) FROM BookingTable WHERE
BookingTable.event_id = EventTable.event_id) AS total_tickets_sold FROM
EventTable;
```

	event_id	event_name	total_tickets_sold
▶	1	Event1	NULL
	2	Event2	NULL
	3	Event3	NULL
	4	Event4	NULL
	5	Event5	NULL

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

    a.
```
SELECT *FROM CustomerTable
WHERE NOT EXISTS (SELECT 1 FROM BookingTable WHERE BookingTable.customer_id = CustomerTable.customer_id);
```



	booking_id	customer_id	customer_name	email	phone_number
*	NULL	NULL	NULL	NULL	NULL

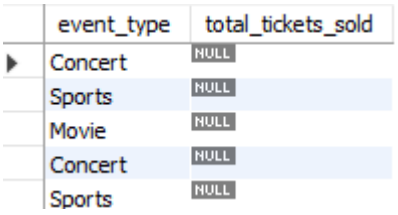
5. *List Events with No Ticket Sales Using a NOT IN Subquery.*

    a.
```
SELECT * FROM EventTable
WHERE event_id NOT IN (SELECT DISTINCT event_id FROM BookingTable);
```

	booking_id	customer_id	customer_name	email	phone_number
*	NULL	NULL	NULL	NULL	NULL

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

    a.
```
SELECT event_type,
(SELECT SUM(CAST(num_tickets AS SIGNED)) FROM BookingTable WHERE BookingTable.event_id = EventTable.event_id) AS total_tickets_sold
FROM EventTable
GROUP BY event_type;
```



	event_type	total_tickets_sold
▶	Concert	NULL
	Sports	NULL
	Movie	NULL
	Concert	NULL
	Sports	NULL

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

    a.
```
SELECT *FROM EventTable
WHERE ticket_price > (SELECT AVG(ticket_price) FROM EventTable);
```

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

a.
```
SELECT customer_id, customer_name,
(SELECT SUM(CAST(total_cost AS DECIMAL(10, 2))) FROM BookingTable WHERE
BookingTable.customer_id = CustomerTable.customer_id) AS total_revenue
FROM CustomerTable;
```

customer_id	customer_name	total_revenue
1001	John Doe	75.00
1002	Jane Smith	50.00
1003	Bob Johnson	100.00
1004	Alice Lee	60.00
1005	Charlie Brown	85.00

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

a.
```
SELECT *FROM CustomerTable
WHERE customer_id IN (SELECT DISTINCT customer_id FROM BookingTable WHERE
BookingTable.event_id IN (SELECT event_id FROM EventTable WHERE venue_id =
1));
```

booking_id	customer_id	customer_name	email	phone_number
1	1001	John Doe	john.doe@example.com	+1234567890
NULL	NULL	NULL	NULL	NULL

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

a.
```
SELECT event_type,
(SELECT SUM(CAST(num_tickets AS SIGNED)) FROM BookingTable WHERE
BookingTable.event_id = EventTable.event_id) AS total_tickets_sold
FROM EventTable
GROUP BY
event_type;
```

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

a.
```
SELECTEventTable.event_type,
COALESCE(SUM(CAST(EventBookings.num_tickets AS SIGNED)), 0) AS
total_tickets_sold
FROM EventTable
LEFT JOIN (SELECTevent_id,
SUM(CAST(num_tickets AS SIGNED)) AS num_tickets
FROM BookingTable
GROUP BY
event_id
) AS EventBookings ON EventTable.event_id = EventBookings.event_id
GROUP BY
EventTable.event_type
LIMIT 0, 1000;
```

event_type	total_tickets_sold
Concert	0
Sports	0
Movie	0

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

a.
```
SELECT venue_id,venue_name,
(SELECT AVG(ticket_price) FROM EventTable WHERE venue_id = venue.venue_id) AS
average_ticket_price
FROM venue;
```

venue_id	venue_name	average_ticket_price
1	Venue1	50.000000
2	Venue2	45.000000
3	Venue3	60.000000
4	Venue4	40.000000
5	Venue5	55.000000