Ticket Booking System

Task 1

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 DATABASE TicketBookingSystem;
USE TicketBookingSystem;
```

1	12:18:25	CREATE DATABASE TicketBookingSystem
2	12:18:25	USE TicketBookingSystem

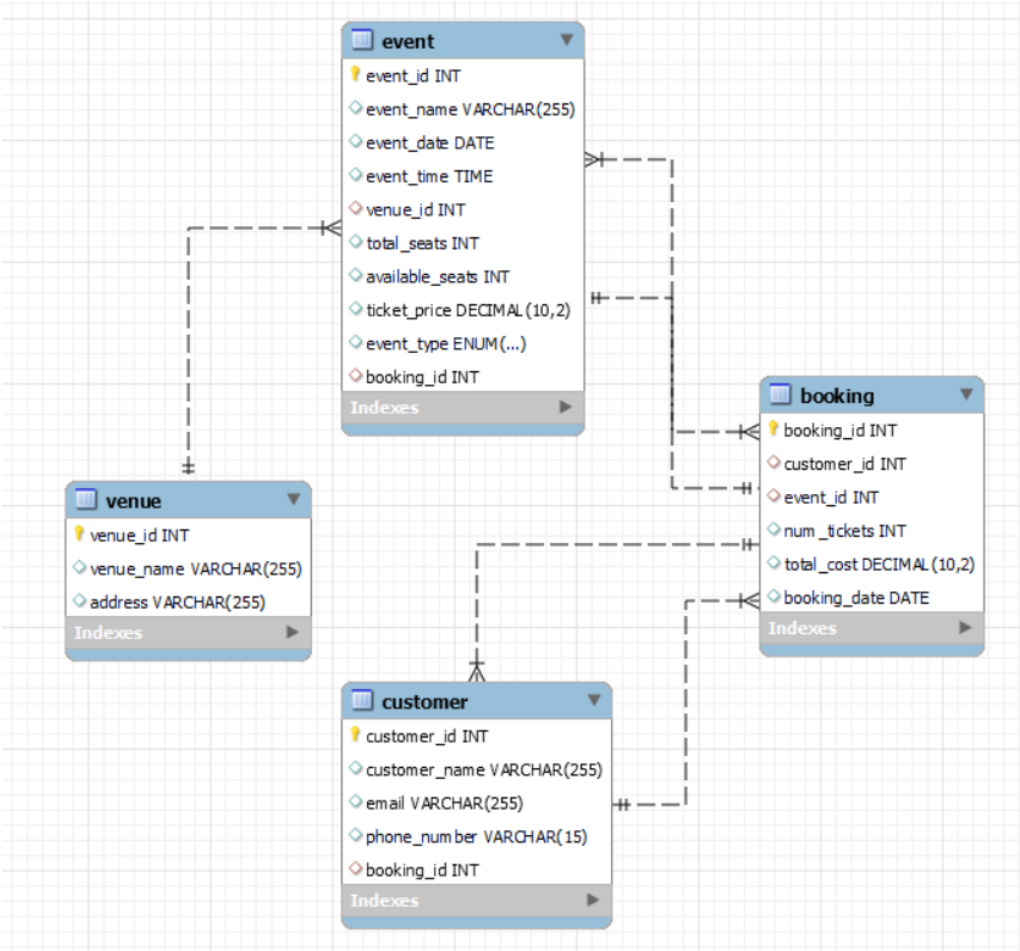
```sql
CREATE TABLE Venue (
    venue_id INT PRIMARY KEY,
    venue_name VARCHAR(255),
    address VARCHAR(255)
);
CREATE TABLE Event (
    event_id INT PRIMARY KEY,
    event_name VARCHAR(255),
    event_date DATE,
    event_time TIME,
    venue_id INT,
    total_seats INT,
    available_seats INT,
    ticket_price DECIMAL(10, 2),
    event_type ENUM('Movie', 'Sports', 'Concert'),
    booking_id INT
);
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(15),
    booking_id INT
);
```

```sql
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY,
    customer_id INT,
    event_id INT,
    num_tickets INT,
    total_cost DECIMAL(10, 2),
    booking_date DATE
);

ALTER TABLE Event
ADD FOREIGN KEY (venue_id) REFERENCES Venue(venue_id),
ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

ALTER TABLE Customer
ADD FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);

ALTER TABLE Booking
ADD FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
ADD 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, 'Theater One', '123 Main Street'),
(2, 'Stadium Arena', '456 Sports Blvd'),
(3, 'Concert Hall', '789 Music Avenue'),
(4, 'Cinema Palace', '101 Movie Lane'),
(5, 'Sports Complex', '202 Game Street');

INSERT INTO Event (event_id, event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type, booking_id)
(1, 'Movie Night', '2023-12-15', '18:00:00', 4, 100, 100, 10.00, 'Movie', 1),
(2, 'Football Match', '2023-12-18', '15:30:00', 2, 50000, 50000, 25.00, 'Sports', 2),
(3, 'Rock Concert', '2023-12-20', '20:00:00', 3, 500, 500, 35.00, 'Concert', 3),
(4, 'Basketball Game', '2023-12-22', '19:00:00', 5, 15000, 15000, 20.00, 'Sports', 4),
(5, 'Drama Play', '2023-12-25', '17:00:00', 1, 200, 200, 15.00, 'Movie', 5),
(6, 'Tennis Tournament', '2023-12-28', '12:00:00', 2, 1000, 1000, 30.00, 'Sports', 6),
(7, 'Jazz Night', '2023-12-30', '21:00:00', 3, 300, 300, 40.00, 'Concert', 7),
(8, 'Cricket Match', '2024-01-02', '14:00:00', 5, 20000, 20000, 25.00, 'Sports', 8),
(9, 'Comedy Show', '2024-01-05', '19:30:00', 4, 150, 150, 18.00, 'Concert', 9),
(10, 'Romantic Movie', '2024-01-08', '20:30:00', 1, 120, 120, 12.00, 'Movie', 10);

INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id) VALUES
(1, 'John Doe', 'john@example.com', '123-456-7890', 1),
(2, 'Alice Smith', 'alice@example.com', '987-654-3210', 2),
(3, 'Bob Johnson', 'bob@example.com', '555-123-4567', 3),
(4, 'Emily Davis', 'emily@example.com', '111-222-3333', 4),
(5, 'Michael White', 'michael@example.com', '444-555-6666', 5),
(6, 'Sophia Miller', 'sophia@example.com', '777-888-9999', 6),
(7, 'Daniel Brown', 'daniel@example.com', '666-777-8888', 7),
(8, 'Olivia Wilson', 'olivia@example.com', '999-000-1111', 8),
(9, 'Matthew Lee', 'matthew@example.com', '222-333-4444', 9),
(10, 'Ava Turner', 'ava@example.com', '333-444-5555', 10);

INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
(1, 1, 1, 2, 20.00, '2023-12-10'),
(2, 2, 2, 4, 100.00, '2023-12-12'),
(3, 3, 3, 1, 35.00, '2023-12-14'),
(4, 4, 4, 3, 60.00, '2023-12-16'),
(5, 5, 5, 2, 30.00, '2023-12-18'),
(6, 6, 6, 5, 150.00, '2023-12-20'),
(7, 7, 7, 1, 40.00, '2023-12-22'),
(8, 8, 8, 4, 100.00, '2023-12-24'),
(9, 9, 9, 3, 54.00, '2023-12-26'),
(10, 10, 10, 2, 24.00, '2023-12-28');
```

2. Write a SQL query to list all Events.

```
SELECT * 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	Movie Night	2023-12-15	18:00:00	4	100	100	10.00	Movie	1
2	Football Match	2023-12-18	15:30:00	2	50000	50000	25.00	Sports	2
3	Rock Concert	2023-12-20	20:00:00	3	500	500	35.00	Concert	3
4	Basketball Game	2023-12-22	19:00:00	5	15000	15000	20.00	Sports	4
5	Drama Play	2023-12-25	17:00:00	1	200	200	15.00	Movie	5
6	Tennis Tournament	2023-12-28	12:00:00	2	1000	1000	30.00	Sports	6
7	Jazz Night	2023-12-30	21:00:00	3	300	300	40.00	Concert	7
8	Cricket Match	2024-01-02	14:00:00	5	20000	20000	25.00	Sports	8
9	Comedy Show	2024-01-05	19:30:00	4	150	150	18.00	Concert	9
10	Romantic Movie	2024-01-08	20:30:00	1	120	120	12.00	Movie	10

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

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

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	Movie Night	2023-12-15	18:00:00	4	100	100	10.00	Movie	1
2	Football Match	2023-12-18	15:30:00	2	50000	50000	25.00	Sports	2
3	Rock Concert	2023-12-20	20:00:00	3	500	500	35.00	Concert	3
4	Basketball Game	2023-12-22	19:00:00	5	15000	15000	20.00	Sports	4
5	Drama Play	2023-12-25	17:00:00	1	200	200	15.00	Movie	5
6	Tennis Tournament	2023-12-28	12:00:00	2	1000	1000	30.00	Sports	6
7	Jazz Night	2023-12-30	21:00:00	3	300	300	40.00	Concert	7
8	Cricket Match	2024-01-02	14:00:00	5	20000	20000	25.00	Sports	8
9	Comedy Show	2024-01-05	19:30:00	4	150	150	18.00	Concert	9
10	Romantic Movie	2024-01-08	20:30:00	1	120	120	12.00	Movie	10

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

```sql
SELECT * FROM Event WHERE event_name LIKE '%cup%';
```

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

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

```sql
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

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

```sql
SELECT * FROM Event WHERE event_date BETWEEN '2023-12-15' AND '2023-12-31';
```

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	Movie Night	2023-12-15	18:00:00	4	100	100	10.00	Movie	1
2	Football Match	2023-12-18	15:30:00	2	50000	50000	25.00	Sports	2
3	Rock Concert	2023-12-20	20:00:00	3	500	500	35.00	Concert	3
4	Basketball Game	2023-12-22	19:00:00	5	15000	15000	20.00	Sports	4
5	Drama Play	2023-12-25	17:00:00	1	200	200	15.00	Movie	5
6	Tennis Tournament	2023-12-28	12:00:00	2	1000	1000	30.00	Sports	6
7	Jazz Night	2023-12-30	21:00:00	3	300	300	40.00	Concert	7

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

```
SELECT * FROM Event WHERE available_seats > 0 AND event_type = '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
3	Rock Concert	2023-12-20	20:00:00	3	500	500	35.00	Concert	3
7	Jazz Night	2023-12-30	21:00:00	3	300	300	40.00	Concert	7
9	Comedy Show	2024-01-05	19:30:00	4	150	150	18.00	Concert	9

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

```
SELECT * FROM Customer LIMIT 5 OFFSET 5;
```

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

customer_id	customer_name	email	phone_number	booking_id
6	Sophia Miller	sophia@example.com	777-888-9999	6
7	Daniel Brown	daniel@example.com	666-777-8888	7
8	Olivia Wilson	olivia@example.com	999-000-1111	8
9	Matthew Lee	matthew@example.com	222-333-4444	9
10	Ava Turner	ava@example.com	333-444-5555	10

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

```
SELECT * FROM Booking WHERE num_tickets > 4;
```

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

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
6	6	6	5	150.00	2023-12-20

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

```sql
SELECT * FROM Customer WHERE phone_number LIKE '%000';
```

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

customer_id	customer_name	email	phone_number	booking_id

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

```sql
SELECT * FROM Event WHERE 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
8	Cricket Match	2024-01-02	14:00:00	5	20000	20000	25.00	Sports	8
2	Football Match	2023-12-18	15:30:00	2	50000	50000	25.00	Sports	2

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

```sql
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	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Movie Night	2023-12-15	18:00:00	4	100	100	10.00	Movie	1
2	Football Match	2023-12-18	15:30:00	2	50000	50000	25.00	Sports	2
3	Rock Concert	2023-12-20	20:00:00	3	500	500	35.00	Concert	3
4	Basketball Game	2023-12-22	19:00:00	5	15000	15000	20.00	Sports	4
5	Drama Play	2023-12-25	17:00:00	1	200	200	15.00	Movie	5
6	Tennis Tournament	2023-12-28	12:00:00	2	1000	1000	30.00	Sports	6
7	Jazz Night	2023-12-30	21:00:00	3	300	300	40.00	Concert	7
8	Cricket Match	2024-01-02	14:00:00	5	20000	20000	25.00	Sports	8
9	Comedy Show	2024-01-05	19:30:00	4	150	150	18.00	Concert	9
10	Romantic Movie	2024-01-08	20:30:00	1	120	120	12.00	Movie	10

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 e.event_id, e.event_name, AVG(e.ticket_price) AS average_ticket_price
FROM Event e
GROUP BY e.event_id, e.event_name;
```

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

event_id	event_name	average_ticket_price
1	Movie Night	10.000000
2	Football Match	25.000000
3	Rock Concert	35.000000
4	Basketball Game	20.000000
5	Drama Play	15.000000
6	Tennis Tournament	30.000000
7	Jazz Night	40.000000
8	Cricket Match	25.000000
9	Comedy Show	18.000000
10	Romantic Movie	12.000000

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

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

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

event_id	event_name	total_revenue
1	Movie Night	20.00
2	Football Match	100.00
3	Rock Concert	35.00
4	Basketball Game	60.00
5	Drama Play	30.00
6	Tennis Tournament	150.00
7	Jazz Night	40.00
8	Cricket Match	100.00
9	Comedy Show	54.00
10	Romantic Movie	24.00

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

```sql
SELECT e.event_id, e.event_name, SUM(b.num_tickets) AS total_tickets_sold
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_id
ORDER BY total_tickets_sold DESC
LIMIT 1;
```

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

event_id	event_name	total_tickets_sold
6	Tennis Tournament	5

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

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

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

event_id	event_name	total_tickets_sold
1	Movie Night	2
2	Football Match	4
3	Rock Concert	1
4	Basketball Game	3
5	Drama Play	2
6	Tennis Tournament	5
7	Jazz Night	1
8	Cricket Match	4
9	Comedy Show	3
10	Romantic Movie	2

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

```sql
SELECT e.event_id, e.event_name
FROM Event e
LEFT JOIN Booking b ON e.event_id = b.event_id
WHERE b.booking_id IS NULL;
```

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

event_id	event_name

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

```sql
SELECT c.customer_id, c.customer_name, SUM(b.num_tickets) AS total_tickets_booked
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
GROUP BY c.customer_id
ORDER BY total_tickets_booked DESC LIMIT 1;
```

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

customer_id	customer_name	total_tickets_booked
6	Sophia Miller	5

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

```sql
SELECT e.event_id, e.event_name, MONTH(b.booking_date) AS month, SUM(b.num_tickets) AS total_tickets_sold
FROM Event e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_id, month
ORDER BY e.event_id, month;
```

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

event_id	event_name	month	total_tickets_sold
1	Movie Night	12	2
2	Football Match	12	4
3	Rock Concert	12	1
4	Basketball Game	12	3
5	Drama Play	12	2
6	Tennis Tournament	12	5
7	Jazz Night	12	1
8	Cricket Match	12	4
9	Comedy Show	12	3
10	Romantic Movie	12	2

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

```sql
SELECT v.venue_id, v.venue_name, AVG(e.ticket_price) AS average_ticket_price
FROM Venue v
JOIN Event e ON v.venue_id = e.venue_id
GROUP BY v.venue_id;
```

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

event_id	event_name	month	total_tickets_sold
1	Movie Night	12	2
2	Football Match	12	4
3	Rock Concert	12	1
4	Basketball Game	12	3
5	Drama Play	12	2
6	Tennis Tournament	12	5
7	Jazz Night	12	1
8	Cricket Match	12	4
9	Comedy Show	12	3
10	Romantic Movie	12	2

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

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

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

event_type	total_tickets_sold
Movie	6
Sports	16
Concert	5

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

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

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

year	total_revenue
2023	613.00

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

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

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

customer_id	customer_name

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

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

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

customer_id	customer_name	total_revenue
1	John Doe	20.00
2	Alice Smith	100.00
3	Bob Johnson	35.00
4	Emily Davis	60.00
5	Michael White	30.00
6	Sophia Miller	150.00
7	Daniel Brown	40.00
8	Olivia Wilson	100.00
9	Matthew Lee	54.00
10	Ava Turner	24.00

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

```sql
SELECT e.event_type, v.venue_id, v.venue_name, AVG(e.ticket_price) AS average_ticket_price
FROM Event e
JOIN Venue v ON e.venue_id = v.venue_id
GROUP BY e.event_type, v.venue_id;
```

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

event_type	venue_id	venue_name	average_ticket_price
Movie	1	Theater One	13.500000
Sports	2	Stadium Arena	27.500000
Concert	3	Concert Hall	37.500000
Movie	4	Cinema Palace	10.000000
Concert	4	Cinema Palace	18.000000
Sports	5	Sports Complex	22.500000

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

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

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

customer_id	customer_name	total_tickets_purchased
1	John Doe	2
2	Alice Smith	4
3	Bob Johnson	1
4	Emily Davis	3
5	Michael White	2
6	Sophia Miller	5
7	Daniel Brown	1
8	Olivia Wilson	4
9	Matthew Lee	3
10	Ava Turner	2

Tasks 4: Subquery and its types

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

```sql
SELECT v.venue_id, v.venue_name,
(SELECT AVG(e.ticket_price) FROM Event e WHERE e.venue_id = v.venue_id) AS average_ticket_price
FROM Venue v;
```

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

venue_id	venue_name	average_ticket_price
1	Theater One	13.500000
2	Stadium Arena	27.500000
3	Concert Hall	37.500000
4	Cinema Palace	14.000000
5	Sports Complex	22.500000

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

```sql
SELECT e.event_id, e.event_name
FROM Event e
WHERE (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id) > 0.5 * e.total_seats;
```

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

event_id	event_name

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

```sql
SELECT e.event_id, e.event_name,
(SELECT SUM(b.num_tickets)
FROM Booking b
WHERE b.event_id = e.event_id) AS "total_tickets_sold"
FROM Event e
```

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

event_id	event_name	total_tickets_sold
1	Movie Night	2
2	Football Match	4
3	Rock Concert	1
4	Basketball Game	3
5	Drama Play	2
6	Tennis Tournament	5
7	Jazz Night	1
8	Cricket Match	4
9	Comedy Show	3
10	Romantic Movie	2

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

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

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

customer_id	customer_name

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

```
SELECT e.event_id, e.event_name
FROM Event e
WHERE e.event_id NOT IN (SELECT event_id FROM Booking b);
```

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

event_id	event_name

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

```
SELECT event_type, SUM(total_tickets_sold) AS total_tickets_sold
FROM (SELECT e.event_type,
    (SELECT SUM(b.num_tickets) FROM Booking b
    WHERE b.event_id = e.event_id) AS total_tickets_sold
    FROM Event e) AS subquery
GROUP BY event_type;
```

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

event_type	total_tickets_sold
Movie	6
Sports	16
Concert	5

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

```
SELECT e.event_id, e.event_name, e.ticket_price
FROM Event e
WHERE e.ticket_price > (SELECT AVG(ticket_price) FROM Event);
```

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

event_id	event_name	ticket_price
2	Football Match	25.00
3	Rock Concert	35.00
6	Tennis Tournament	30.00
7	Jazz Night	40.00
8	Cricket Match	25.00

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

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

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

customer_id	customer_name	total_revenue
1	John Doe	20.00
2	Alice Smith	100.00
3	Bob Johnson	35.00
4	Emily Davis	60.00
5	Michael White	30.00
6	Sophia Miller	150.00
7	Daniel Brown	40.00
8	Olivia Wilson	100.00
9	Matthew Lee	54.00
10	Ava Turner	24.00

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

```sql
SELECT c.customer_id, c.customer_name
FROM Customer c
WHERE
    EXISTS (SELECT 1 FROM Booking b
    JOIN Event e ON b.event_id = e.event_id
    WHERE e.venue_id = 1 AND b.customer_id = c.customer_id);
```

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

customer_id	customer_name
5	Michael White
10	Ava Turner

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

```sql
SELECT event_type, SUM(booking_num_tickets) AS Tickets_Booked FROM
(SELECT e.event_type,
(SELECT b.num_tickets FROM Booking b
WHERE e.event_id = b.event_id)
AS booking_num_tickets FROM Event e) AS subquery
GROUP BY event_type;
```

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

event_type	Tickets_Booked
Movie	6
Sports	16
Concert	5

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

```sql
SELECT MONTH(b.booking_date) AS "Month",
(SELECT COUNT(c.customer_id) FROM Customer c
WHERE c.customer_id = b.customer_id) AS "NoOfBooking"
FROM Booking b;
```

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

MONTH(b.booking_date)	NoOfBooking
12	1
12	1
12	1
12	1
12	1
12	1
12	1
12	1
12	1
12	1

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

```sql
SELECT v.venue_id, v.venue_name,
    (SELECT AVG(e.ticket_price) FROM Event e
    WHERE e.venue_id = v.venue_id) AS average_ticket_price
FROM Venue v;
```

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

venue_id	venue_name	average_ticket_price
1	Theater One	13.500000
2	Stadium Arena	27.500000
3	Concert Hall	37.500000
4	Cinema Palace	14.000000
5	Sports Complex	22.500000