ASSIGNMENT

Ticket Booking System

NAME : PRINCE PATEL

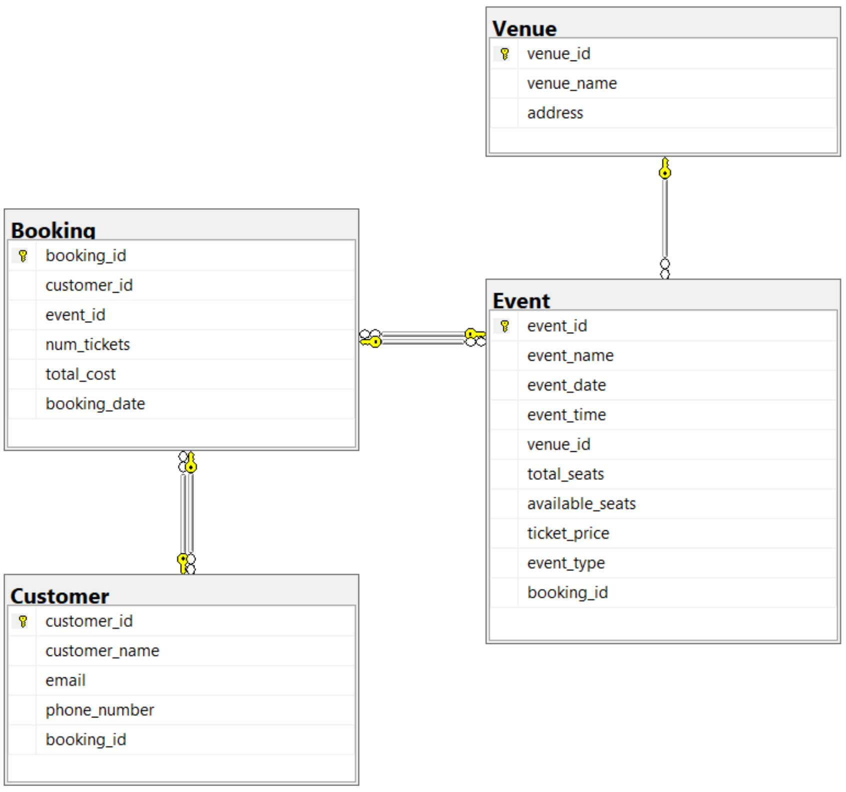
ASSIGNMENT : Ticket Booking System

```
-------------TASK 1: Database Design:

--1. Create the database named "TicketBookingSystem"

CREATE DATABASE TicketBookingSystem;
USE TicketBookingSystem;

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

/*Venu Table
• venue_id (Primary Key)
• venue_name,
• address */

CREATE TABLE Venue (
        venue_id INT PRIMARY KEY,
        venue_name VARCHAR(284),
        address VARCHAR(284)
);

/*Event Table
• event_id (Primary Key)
```

```sql
• 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) */

CREATE TABLE Event (
    event_id INT PRIMARY KEY,
    event_name VARCHAR(180),
    event_date DATE,
    event_time TIME,
    venue_id INT,
    total_seats INT,
    available_seats INT,
    ticket_price DECIMAL(10, 2),
    event_type VARCHAR(70),
    booking_id INT,     --- Foreign key will be declared after table creation
    CONSTRAINT FK_Event_Venue FOREIGN KEY (venue_id) REFERENCES Venue(venue_id)
);

/*Customer Table
• customer_id (Primary key)
• customer_name,
• email,
• phone_number,
• booking_id (Foreign Key) */
```

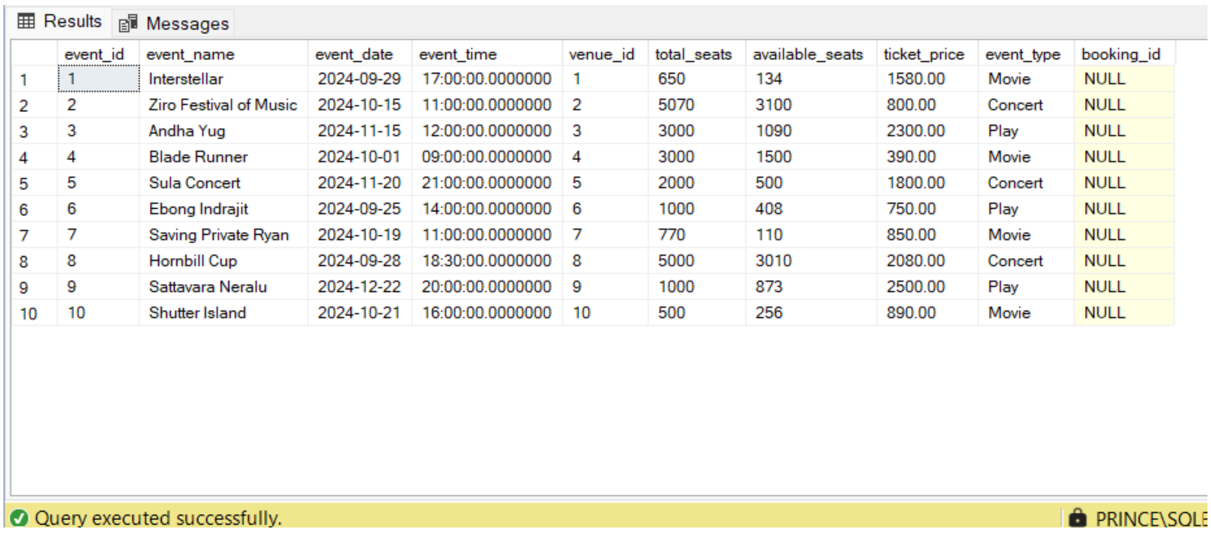
```
CREATE TABLE Customer (
            customer_id INT PRIMARY KEY,
            customer_name VARCHAR(180),
            email VARCHAR(284),
            phone_number VARCHAR(20),
            booking_id INT      --- Foreign key will be declared after table creation
);

/*Booking Table
• booking_id (Primary Key),
• customer_id (Foreign Key),
• event_id (Foreign Key),
• num_tickets,
• total_cost,
• booking_date */

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,
            CONSTRAINT FK_Booking_Customer FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id),
            CONSTRAINT FK_Booking_Event FOREIGN KEY (event_id) REFERENCES Event(event_id)
);

ALTER TABLE Event
ADD CONSTRAINT FK_Event_Booking FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
```

```
ALTER TABLE Customer

ADD CONSTRAINT FK_Customer_Booking FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);


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



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


-- All primary and foreign keys are inserted while creating the table.


-------------------------------------------------------------------------------------------

------------TASK 2: Select, Where, Between, AND, LIKE:


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


INSERT INTO Venue (Venue_id, venue_name, address)
```

```sql
VALUES
(1, 'Raj Mandir Theatre', 'Jaipur'),
(2, 'Jawaharlal Nehru Stadium', 'Delhi'),
(3, 'National Centre for the Performing Arts', 'Mumbai'),
(4, 'Maratha Mandir', 'Mumbai'),
(5, 'Nehru Park', 'Delhi'),
(6, 'LTG Auditorium', 'Delhi'),
(7, 'Prasad IMAX', 'Hyderabad'),
(8, 'Princeton Club', 'Kolkata'),
(9, 'Rabindra Sadan', 'Kolkata'),
(10, 'Minerva Theatre', 'Mumbai');

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, 'Interstellar', '2024-09-29', '17:00:00', 1, 650, 134, 1580.00, 'Movie', NULL),
(2, 'Ziro Festival of Music', '2024-10-15', '11:00:00', 2, 5070, 3100, 800.00, 'Concert', NULL),
(3, 'Andha Yug', '2024-11-15', '12:00:00', 3, 3000, 1090, 2300.00, 'Play', NULL),
(4, 'Blade Runner', '2024-10-01', '09:00:00', 4, 3000, 1500, 390.00, 'Movie', NULL),
(5, 'Sula Concert', '2024-11-20', '21:00:00', 5, 2000, 500, 1800.00, 'Concert', NULL),
(6, 'Ebong Indrajit', '2024-09-25', '14:00:00', 6, 1000, 408, 750.00, 'Play', NULL),
(7, 'Saving Private Ryan', '2024-10-19', '11:00:00', 7, 770, 110, 850.00, 'Movie', NULL),
(8, 'Hornbill Cup', '2024-09-28', '18:30:00', 8, 5000, 3010, 2080.00, 'Concert', NULL),
(9, 'Sattavara Neralu', '2024-12-22', '20:00:00', 9, 1000, 873, 2500.00, 'Play', NULL),
(10, 'Shutter Island', '2024-10-21', '16:00:00', 10, 500, 256, 890.00, 'Movie', NULL);

INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id)
VALUES
(1, 'Prince Patel', 'resonance443731@gmail.com', '8127799219', NULL),
(2, 'Ajay Kumar', 'ajay@gmail.com', '9223799219', NULL),
(3, 'Nisha Gupta', 'nisha@gmail.com', '7823796120', NULL),
(4, 'Piyush Verma', 'piyush@gmail.com', '8432796479', NULL),
(5, 'Karan Prasad', 'karan@gmail.com', '9889787470', NULL),
(6, 'Khushi Pandey', 'khushi@gmail.com', '9244294000', NULL),
(7, 'Manoj Kumar', 'manoj@gmail.com', '8840298385', NULL),
(8, 'Abhishek Singh', 'abhi@gmail.com', '8901230000', NULL),
(9, 'Kamala Yadav', 'kamala@gmail.com', '9012440007', NULL),
(10, 'Tanya Pandey', 'tanya@gmail.com', '8810498582', NULL);

INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date)
VALUES
(1, 1, 1, 2, 3160.00, '2024-09-10'),
(2, 2, 2, 4, 3200.00, '2024-10-05'),
(3, 3, 3, 1, 2300.00, '2024-11-08'),
(4, 4, 4, 3, 1170.00, '2024-09-07'),
(5, 5, 5, 4, 7200.00, '2024-10-10'),
(6, 6, 6, 2, 1500.00, '2024-09-09'),
(7, 7, 7, 4, 3400.00, '2024-10-09'),
(8, 8, 8, 3, 6240.00, '2024-09-11'),
(9, 9, 9, 5, 12500.00, '2024-12-06'),
(10, 10, 10, 3, 2670.00, '2024-09-13');

---2. Write a SQL query to list all Events

SELECT * FROM Event;
```



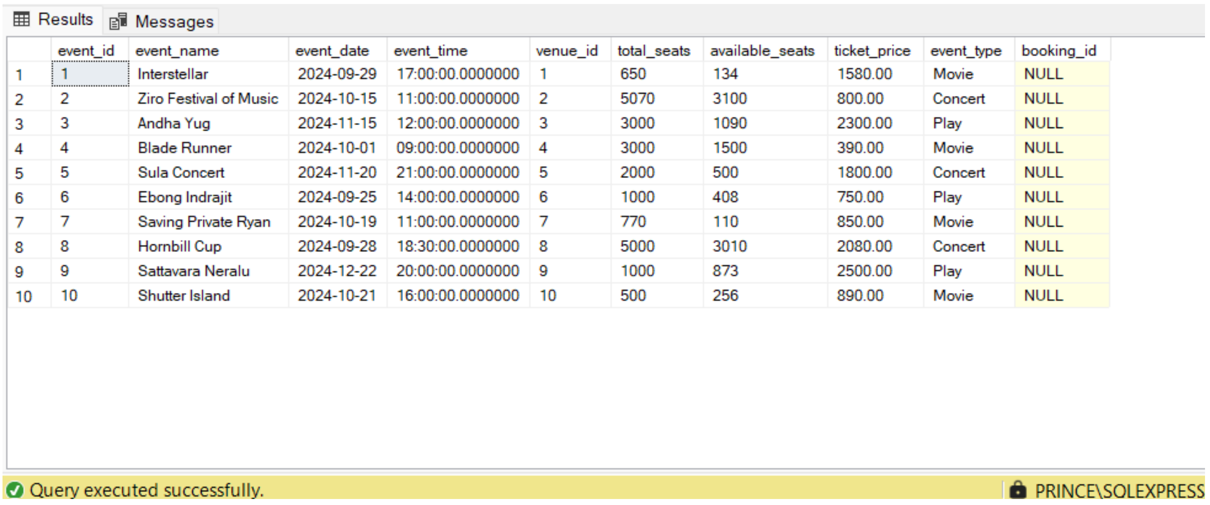
Results | Messages

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	1	Interstellar	2024-09-29	17:00:00.0000000	1	650	134	1580.00	Movie	NULL
2	2	Ziro Festival of Music	2024-10-15	11:00:00.0000000	2	5070	3100	800.00	Concert	NULL
3	3	Andha Yug	2024-11-15	12:00:00.0000000	3	3000	1090	2300.00	Play	NULL
4	4	Blade Runner	2024-10-01	09:00:00.0000000	4	3000	1500	390.00	Movie	NULL
5	5	Sula Concert	2024-11-20	21:00:00.0000000	5	2000	500	1800.00	Concert	NULL
6	6	Ebong Indrajit	2024-09-25	14:00:00.0000000	6	1000	408	750.00	Play	NULL
7	7	Saving Private Ryan	2024-10-19	11:00:00.0000000	7	770	110	850.00	Movie	NULL
8	8	Hornbill Cup	2024-09-28	18:30:00.0000000	8	5000	3010	2080.00	Concert	NULL
9	9	Sattavara Neralu	2024-12-22	20:00:00.0000000	9	1000	873	2500.00	Play	NULL
10	10	Shutter Island	2024-10-21	16:00:00.0000000	10	500	256	890.00	Movie	NULL

Query executed successfully. PRINCE\SQLE

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

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



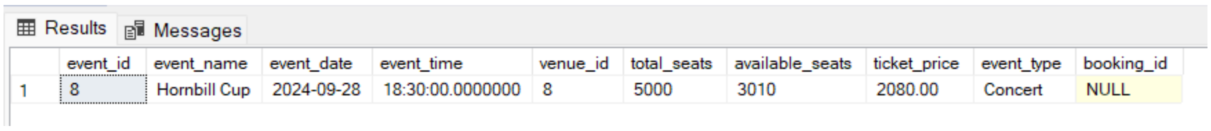
Results | Messages

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	1	Interstellar	2024-09-29	17:00:00.0000000	1	650	134	1580.00	Movie	NULL
2	2	Ziro Festival of Music	2024-10-15	11:00:00.0000000	2	5070	3100	800.00	Concert	NULL
3	3	Andha Yug	2024-11-15	12:00:00.0000000	3	3000	1090	2300.00	Play	NULL
4	4	Blade Runner	2024-10-01	09:00:00.0000000	4	3000	1500	390.00	Movie	NULL
5	5	Sula Concert	2024-11-20	21:00:00.0000000	5	2000	500	1800.00	Concert	NULL
6	6	Ebong Indrajit	2024-09-25	14:00:00.0000000	6	1000	408	750.00	Play	NULL
7	7	Saving Private Ryan	2024-10-19	11:00:00.0000000	7	770	110	850.00	Movie	NULL
8	8	Hornbill Cup	2024-09-28	18:30:00.0000000	8	5000	3010	2080.00	Concert	NULL
9	9	Sattavara Neralu	2024-12-22	20:00:00.0000000	9	1000	873	2500.00	Play	NULL
10	10	Shutter Island	2024-10-21	16:00:00.0000000	10	500	256	890.00	Movie	NULL

Query executed successfully. PRINCE\SQLEXPRESS

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

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

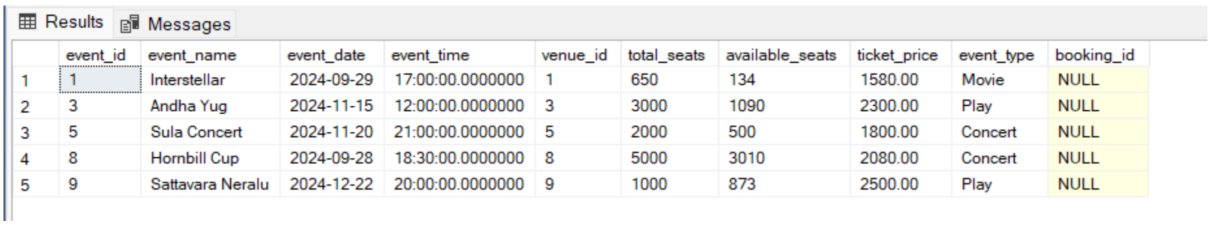


Results | Messages

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	8	Hornbill Cup	2024-09-28	18:30:00.0000000	8	5000	3010	2080.00	Concert	NULL

--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;
```

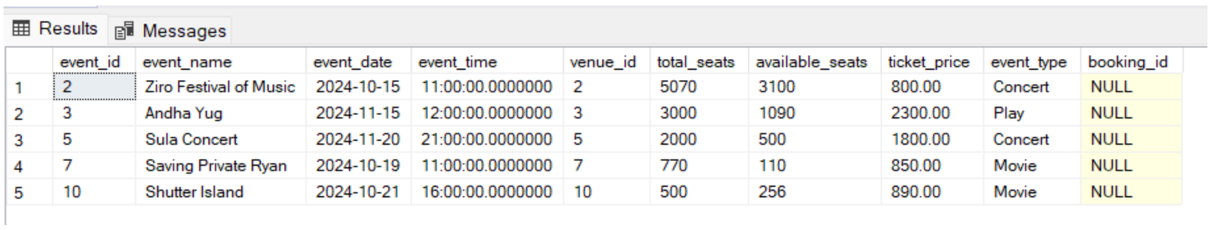


Results | Messages

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	1	Interstellar	2024-09-29	17:00:00.0000000	1	650	134	1580.00	Movie	NULL
2	3	Andha Yug	2024-11-15	12:00:00.0000000	3	3000	1090	2300.00	Play	NULL
3	5	Sula Concert	2024-11-20	21:00:00.0000000	5	2000	500	1800.00	Concert	NULL
4	8	Hornbill Cup	2024-09-28	18:30:00.0000000	8	5000	3010	2080.00	Concert	NULL
5	9	Sattavara Neralu	2024-12-22	20:00:00.0000000	9	1000	873	2500.00	Play	NULL

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

```
SELECT *

FROM Event

WHERE event_date BETWEEN '2024-10-12' AND '2024-11-30';
```

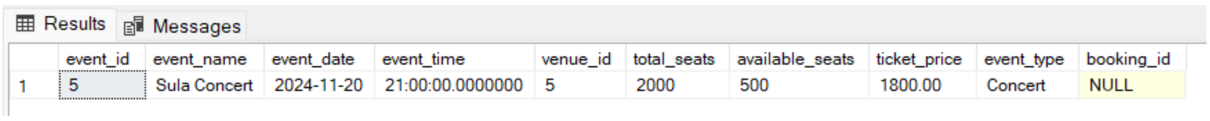


Results | Messages

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	2	Ziro Festival of Music	2024-10-15	11:00:00.0000000	2	5070	3100	800.00	Concert	NULL
2	3	Andha Yug	2024-11-15	12:00:00.0000000	3	3000	1090	2300.00	Play	NULL
3	5	Sula Concert	2024-11-20	21:00:00.0000000	5	2000	500	1800.00	Concert	NULL
4	7	Saving Private Ryan	2024-10-19	11:00:00.0000000	7	770	110	850.00	Movie	NULL
5	10	Shutter Island	2024-10-21	16:00:00.0000000	10	500	256	890.00	Movie	NULL

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

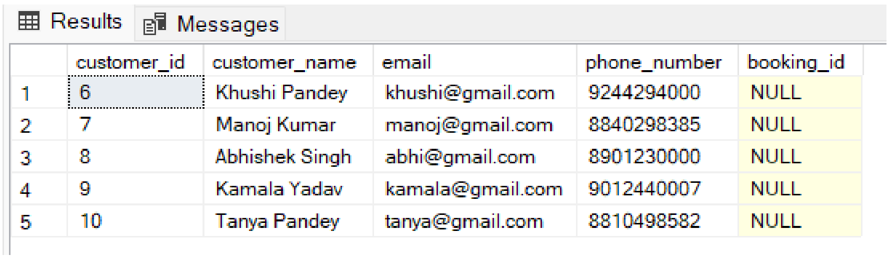
```
SELECT *

FROM Event

WHERE available_seats > 0 AND event_name LIKE '%Concert%';
```


Results | Messages

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	5	Sula Concert	2024-11-20	21:00:00.0000000	5	2000	500	1800.00	Concert	NULL

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

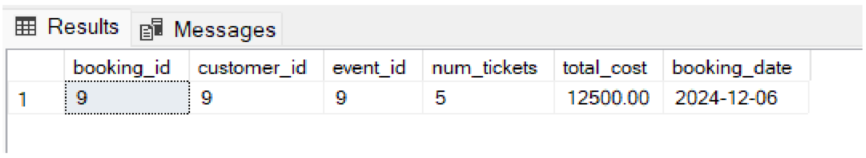
```
SELECT *
FROM Customer
ORDER BY customer_id
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```


Results | Messages

	customer_id	customer_name	email	phone_number	booking_id
1	6	Khushi Pandey	khushi@gmail.com	9244294000	NULL
2	7	Manoj Kumar	manoj@gmail.com	8840298385	NULL
3	8	Abhishek Singh	abhi@gmail.com	8901230000	NULL
4	9	Kamala Yadav	kamala@gmail.com	9012440007	NULL
5	10	Tanya Pandey	tanya@gmail.com	8810498582	NULL

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

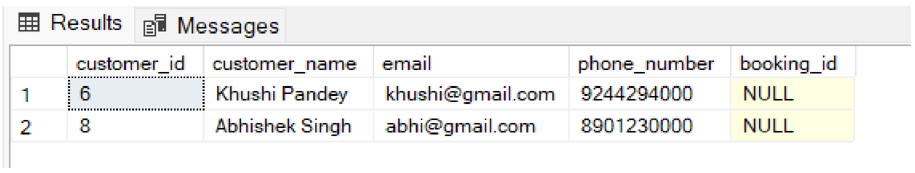
```
SELECT *
FROM BOOKING
WHERE num_tickets > 4;
```


Results | Messages

	booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1	9	9	9	5	12500.00	2024-12-06

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

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

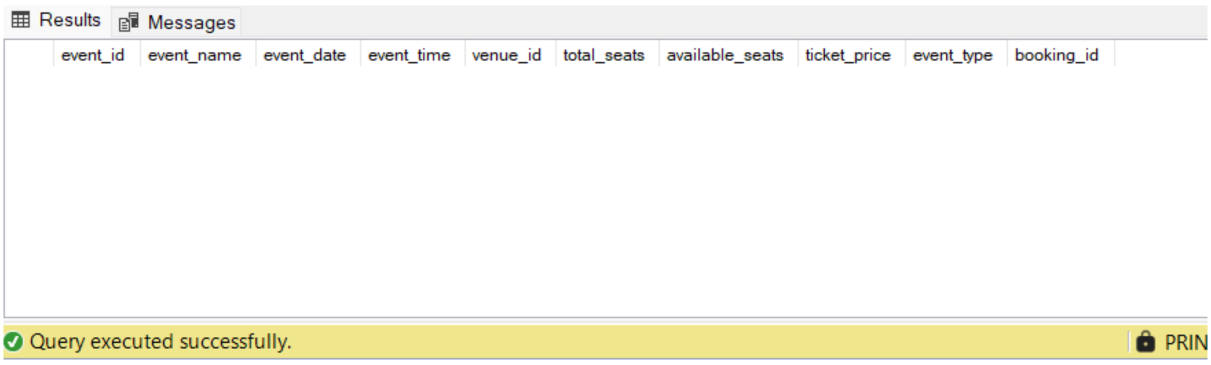


Results | Messages

	customer_id	customer_name	email	phone_number	booking_id
1	6	Khushi Pandey	khushi@gmail.com	9244294000	NULL
2	8	Abhishek Singh	abhi@gmail.com	8901230000	NULL

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

```
SELECT *
FROM Event
WHERE total_seats > 15000
ORDER BY total_seats;
```



Results | Messages

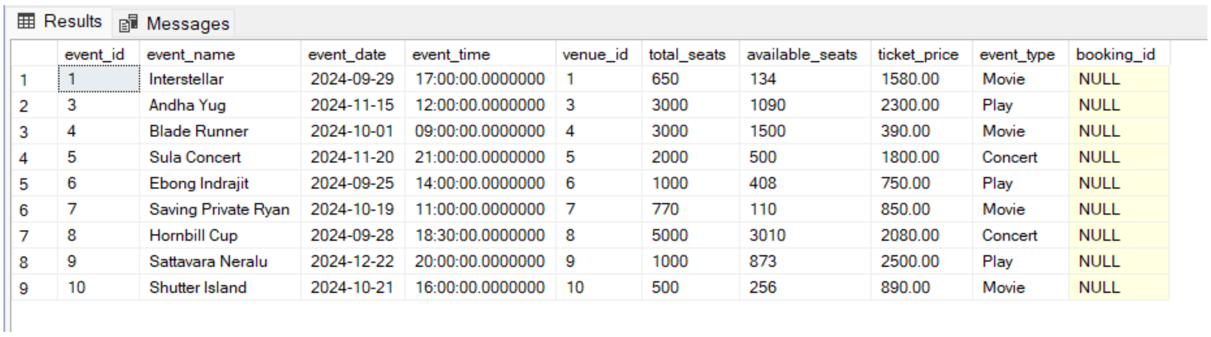
event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id

Query executed successfully. PRIN

/* We are getting no output as there is no event whose seat capacity is more than 15000 */

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

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

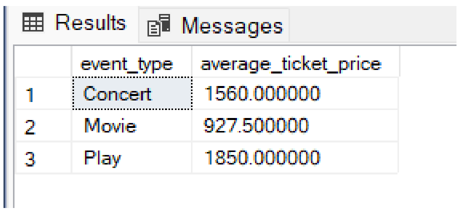

Results | Messages

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	1	Interstellar	2024-09-29	17:00:00.0000000	1	650	134	1580.00	Movie	NULL
2	3	Andha Yug	2024-11-15	12:00:00.0000000	3	3000	1090	2300.00	Play	NULL
3	4	Blade Runner	2024-10-01	09:00:00.0000000	4	3000	1500	390.00	Movie	NULL
4	5	Sula Concert	2024-11-20	21:00:00.0000000	5	2000	500	1800.00	Concert	NULL
5	6	Ebong Indrajit	2024-09-25	14:00:00.0000000	6	1000	408	750.00	Play	NULL
6	7	Saving Private Ryan	2024-10-19	11:00:00.0000000	7	770	110	850.00	Movie	NULL
7	8	Hornbill Cup	2024-09-28	18:30:00.0000000	8	5000	3010	2080.00	Concert	NULL
8	9	Sattavara Neralu	2024-12-22	20:00:00.0000000	9	1000	873	2500.00	Play	NULL
9	10	Shutter Island	2024-10-21	16:00:00.0000000	10	500	256	890.00	Movie	NULL

------------TASK 3: Aggregate functions, Having, Order By, GroupBy and Joins:

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

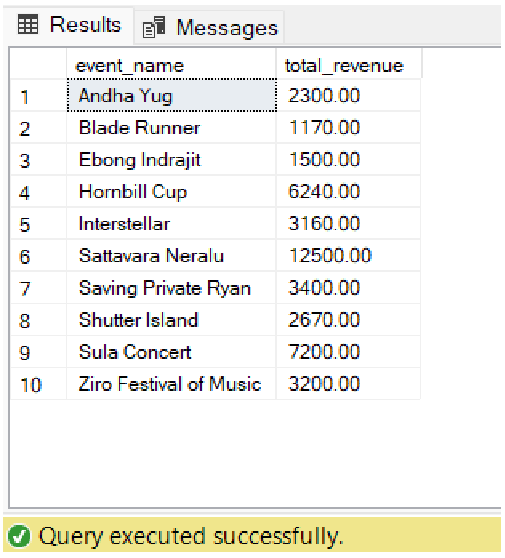
```
SELECT event_type, AVG(ticket_price) AS average_ticket_price

FROM Event

GROUP BY event_type;
```


Results | Messages

	event_type	average_ticket_price
1	Concert	1560.000000
2	Movie	927.500000
3	Play	1850.000000

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

```
SELECT 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_name;
```



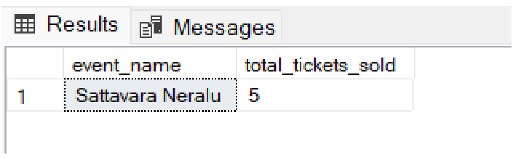
Results | Messages

	event_name	total_revenue
1	Andha Yug	2300.00
2	Blade Runner	1170.00
3	Ebong Indrajit	1500.00
4	Hornbill Cup	6240.00
5	Interstellar	3160.00
6	Sattavara Neralu	12500.00
7	Saving Private Ryan	3400.00
8	Shutter Island	2670.00
9	Sula Concert	7200.00
10	Ziro Festival of Music	3200.00

Query executed successfully.

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

```
SELECT TOP 1 e.event_name, SUM(b.num_tickets) AS total_tickets_sold

FROM Event e

INNER JOIN Booking b ON b.event_id = e.event_id

GROUP BY e.event_name

ORDER BY total_tickets_sold DESC;
```

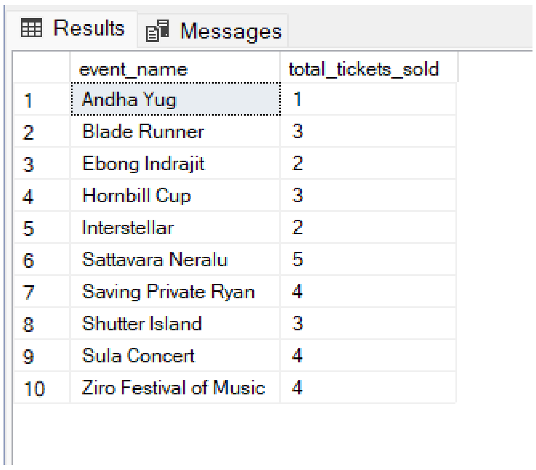


Results | Messages

	event_name	total_tickets_sold
1	Sattavara Neralu	5

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

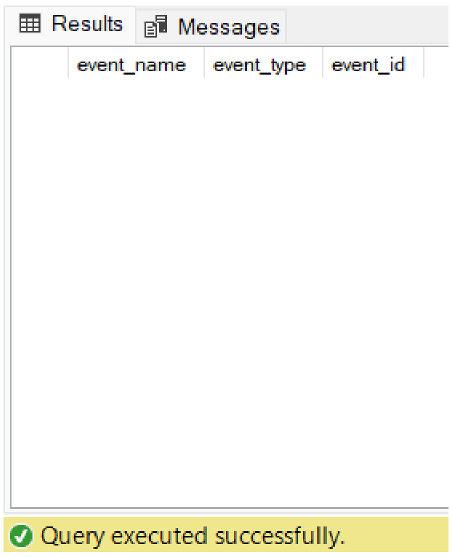
```
SELECT e.event_name, SUM(b.num_tickets) AS total_tickets_sold

FROM Event e

JOIN Booking b ON b.event_id = e.event_id

GROUP BY e.event_name;
```



	event_name	total_tickets_sold
1	Andha Yug	1
2	Blade Runner	3
3	Ebong Indrajit	2
4	Hornbill Cup	3
5	Interstellar	2
6	Sattavara Neralu	5
7	Saving Private Ryan	4
8	Shutter Island	3
9	Sula Concert	4
10	Ziro Festival of Music	4

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

```sql
SELECT e.event_name, e.event_type, e.event_id

FROM Event e

JOIN Booking b ON b.event_id = e.event_id

WHERE b.num_tickets IS NULL;
```



	event_name	event_type	event_id

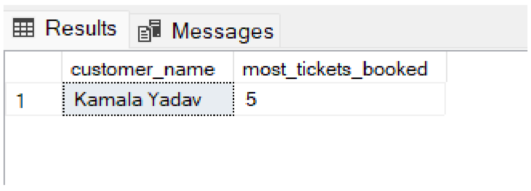
Query executed successfully.

/* We are not getting any output because there are no events with zero ticket sales. */

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

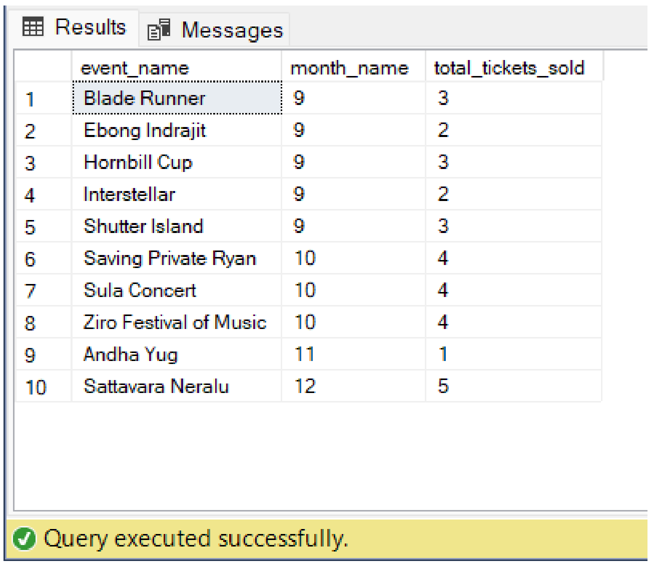
```sql
SELECT TOP 1 c.customer_name, SUM(b.num_tickets) AS most_tickets_booked
```

```
FROM Customer c

INNER JOIN Booking b ON b.booking_id = c.customer_id

GROUP BY c.customer_name

ORDER BY most_tickets_booked DESC;
```


Results Messages

	customer_name	most_tickets_booked
1	Kamala Yadav	5

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

SELECT e.event_name, MONTH(b.booking_date) AS month_name, SUM(b.num_tickets) AS
total_tickets_sold

FROM Event e

JOIN Booking b ON e.event_id = b.booking_id

GROUP BY e.event_name, MONTH(b.booking_date);
```

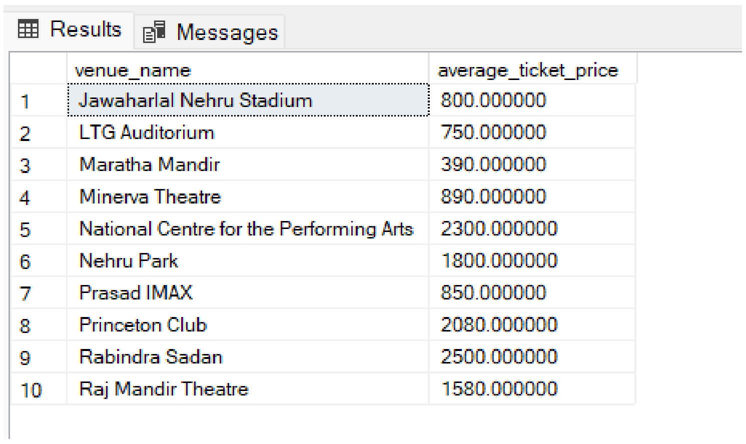

Results Messages

	event_name	month_name	total_tickets_sold
1	Blade Runner	9	3
2	Ebong Indrajit	9	2
3	Hornbill Cup	9	3
4	Interstellar	9	2
5	Shutter Island	9	3
6	Saving Private Ryan	10	4
7	Sula Concert	10	4
8	Ziro Festival of Music	10	4
9	Andha Yug	11	1
10	Sattavara Neralu	12	5

Query executed successfully.

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

SELECT v.venue_name, AVG(e.ticket_price) AS average_ticket_price
```

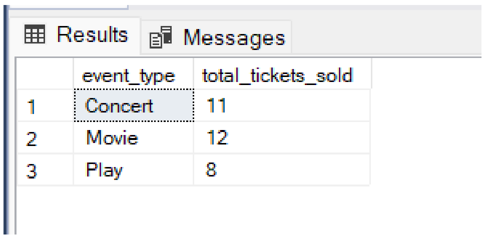
```sql
FROM Event e

INNER JOIN Venue v ON v.venue_id = e.event_id

GROUP BY v.venue_name;
```



Results | Messages

	venue_name	average_ticket_price
1	Jawaharlal Nehru Stadium	800.000000
2	LTG Auditorium	750.000000
3	Maratha Mandir	390.000000
4	Minerva Theatre	890.000000
5	National Centre for the Performing Arts	2300.000000
6	Nehru Park	1800.000000
7	Prasad IMAX	850.000000
8	Princeton Club	2080.000000
9	Rabindra Sadan	2500.000000
10	Raj Mandir Theatre	1580.000000

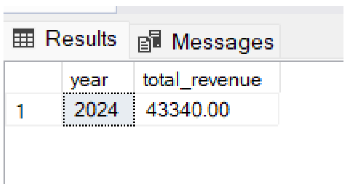
```sql
--9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type

SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold

FROM Event e

JOIN Booking b ON b.booking_id = e.event_id

GROUP BY e.event_type;
```



Results | Messages

	event_type	total_tickets_sold
1	Concert	11
2	Movie	12
3	Play	8

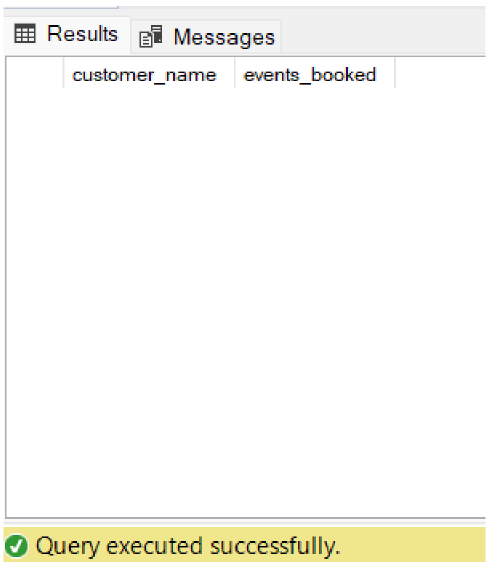
```sql
--10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year

SELECT YEAR(b.booking_date) AS year, SUM(b.total_cost) AS total_revenue

FROM Booking b

JOIN Event e ON b.event_id = e.event_id

GROUP BY YEAR(b.booking_date);
```



```
   year   total_revenue
1  2024   43340.00
```

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

```sql
SELECT c.customer_name, COUNT(DISTINCT b.event_id) AS events_booked

FROM Customer c

JOIN Booking b ON c.customer_id = b.customer_id

GROUP BY c.customer_name

HAVING COUNT(DISTINCT b.event_id) > 1;
```



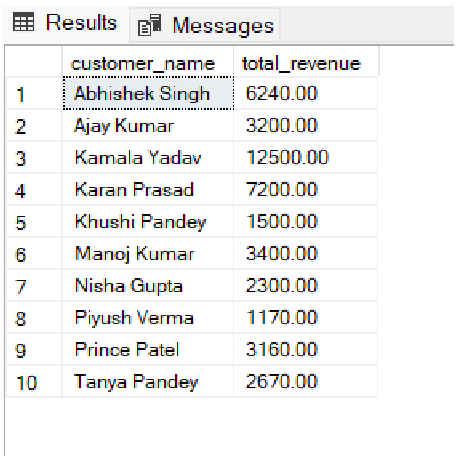
```
   customer_name   events_booked
```

Query executed successfully.

/* We are not getting any output because there are no users who have booked tickets for multiple events. */

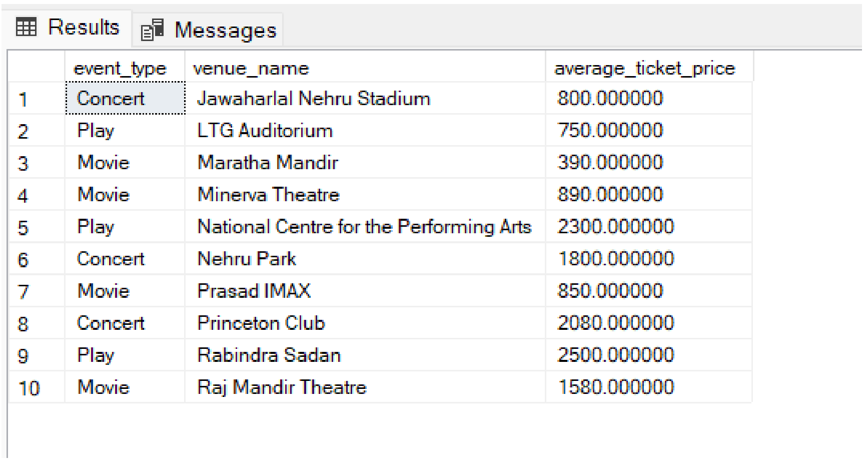
--12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User

```sql
SELECT 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_name;
```


Results | Messages

	customer_name	total_revenue
1	Abhishek Singh	6240.00
2	Ajay Kumar	3200.00
3	Kamala Yadav	12500.00
4	Karan Prasad	7200.00
5	Khushi Pandey	1500.00
6	Manoj Kumar	3400.00
7	Nisha Gupta	2300.00
8	Piyush Verma	1170.00
9	Prince Patel	3160.00
10	Tanya Pandey	2670.00

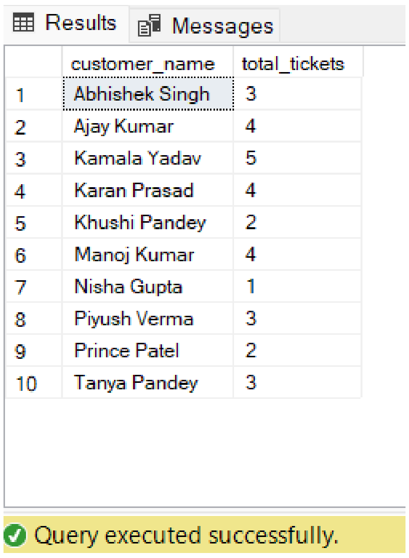
```
--13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue

SELECT e.event_type, 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_name;
```


Results | Messages

	event_type	venue_name	average_ticket_price
1	Concert	Jawaharlal Nehru Stadium	800.000000
2	Play	LTG Auditorium	750.000000
3	Movie	Maratha Mandir	390.000000
4	Movie	Minerva Theatre	890.000000
5	Play	National Centre for the Performing Arts	2300.000000
6	Concert	Nehru Park	1800.000000
7	Movie	Prasad IMAX	850.000000
8	Concert	Princeton Club	2080.000000
9	Play	Rabindra Sadan	2500.000000
10	Movie	Raj Mandir Theatre	1580.000000

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

SELECT c.customer_name, SUM(b.num_tickets) AS total_tickets
FROM Customer c
```

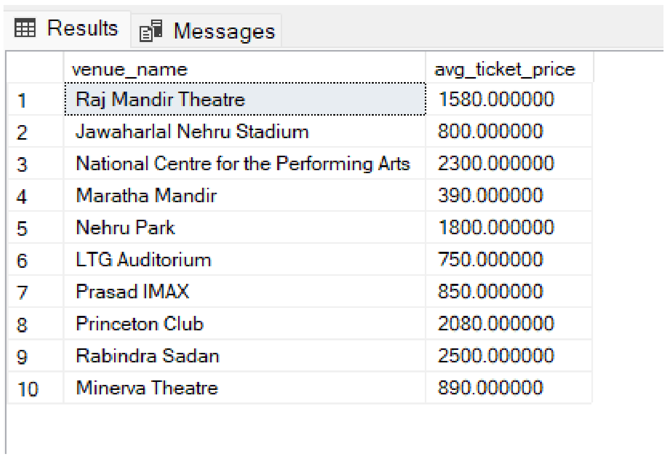
```
JOIN Booking b ON c.customer_id = b.customer_id
WHERE b.booking_date >= DATEADD(DAY, -30, GETDATE())
GROUP BY c.customer_name;
```



	customer_name	total_tickets
1	Abhishek Singh	3
2	Ajay Kumar	4
3	Kamala Yadav	5
4	Karan Prasad	4
5	Khushi Pandey	2
6	Manoj Kumar	4
7	Nisha Gupta	1
8	Piyush Verma	3
9	Prince Patel	2
10	Tanya Pandey	3

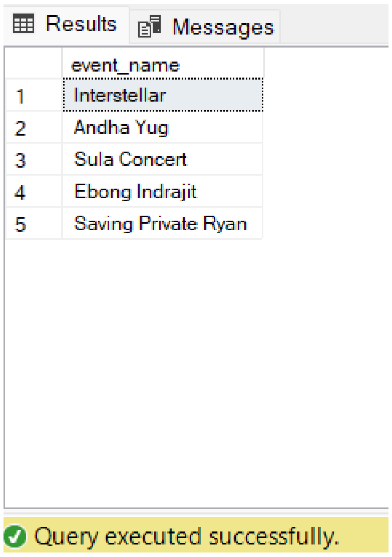
Query executed successfully.

```
-------------------------------------------------------------------------------------------
------------TASK 4: Subquery and its types:

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

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


Results | Messages

	venue_name	avg_ticket_price
1	Raj Mandir Theatre	1580.000000
2	Jawaharlal Nehru Stadium	800.000000
3	National Centre for the Performing Arts	2300.000000
4	Maratha Mandir	390.000000
5	Nehru Park	1800.000000
6	LTG Auditorium	750.000000
7	Prasad IMAX	850.000000
8	Princeton Club	2080.000000
9	Rabindra Sadan	2500.000000
10	Minerva Theatre	890.000000

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

SELECT event_name

FROM Event

WHERE total_seats - available_seats > (SELECT total_seats * 0.50 FROM Event e WHERE e.event_id =
Event.event_id);
```

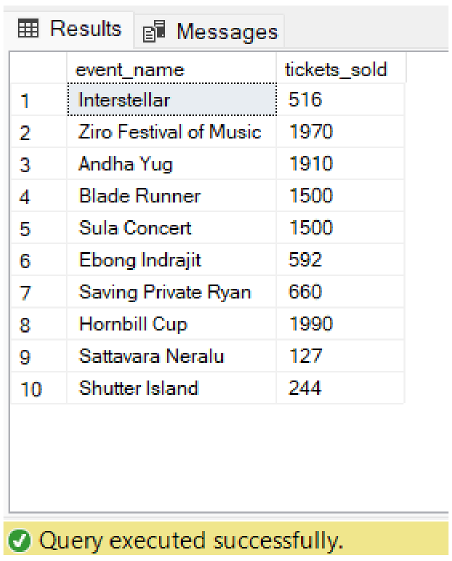

Results | Messages

	event_name
1	Interstellar
2	Andha Yug
3	Sula Concert
4	Ebong Indrajit
5	Saving Private Ryan

Query executed successfully.

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

SELECT event_name,

       (SELECT total_seats - available_seats

        FROM Event e
```

```
    WHERE e.event_id = Event.event_id) AS tickets_sold
FROM Event;
```

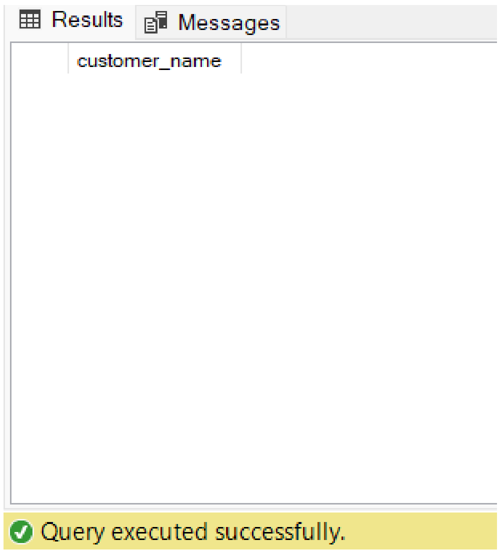


Results | Messages

	event_name	tickets_sold
1	Interstellar	516
2	Ziro Festival of Music	1970
3	Andha Yug	1910
4	Blade Runner	1500
5	Sula Concert	1500
6	Ebong Indrajit	592
7	Saving Private Ryan	660
8	Hornbill Cup	1990
9	Sattavara Neralu	127
10	Shutter Island	244

Query executed successfully.

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

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



Results | Messages

```
customer_name
```

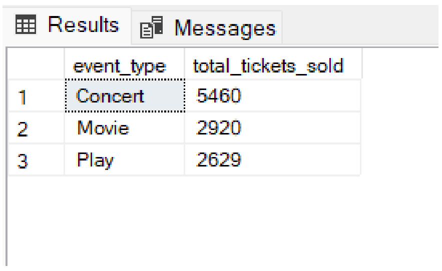
Query executed successfully.

```
/* We are not getting any output because there are no users who haven't booked tickets. */

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

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

```
event_name
```

```
Query executed successfully.
```

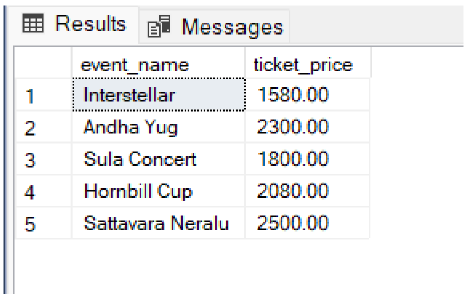
```
/* We are not getting any output because tickets have been sold for all the events. */

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

SELECT e.event_type,
      SUM(sub.tickets_sold) AS total_tickets_sold
FROM Event e,
    (SELECT event_id,
           (total_seats - available_seats) AS tickets_sold
     FROM Event) sub
WHERE e.event_id = sub.event_id
GROUP BY e.event_type;
```



	event_type	total_tickets_sold
1	Concert	5460
2	Movie	2920
3	Play	2629

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

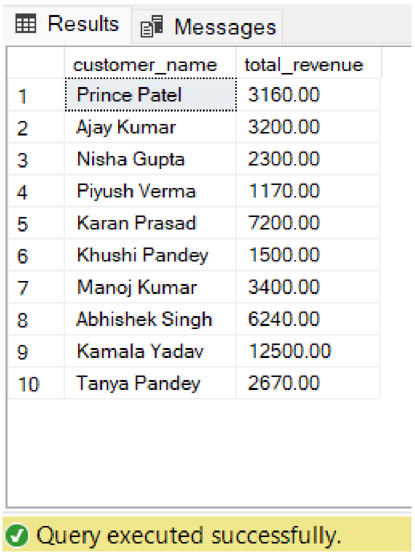
```
SELECT event_name, ticket_price

FROM Event

WHERE ticket_price > (SELECT AVG(ticket_price)

                    FROM Event);
```



	event_name	ticket_price
1	Interstellar	1580.00
2	Andha Yug	2300.00
3	Sula Concert	1800.00
4	Hornbill Cup	2080.00
5	Sattavara Neralu	2500.00

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

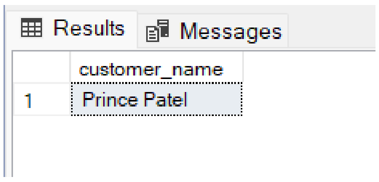
```
SELECT customer_name,

      (SELECT SUM(b.total_cost)

       FROM Booking b

       WHERE b.customer_id = c.customer_id) AS total_revenue

FROM Customer c;
```


Results Messages

	customer_name	total_revenue
1	Prince Patel	3160.00
2	Ajay Kumar	3200.00
3	Nisha Gupta	2300.00
4	Piyush Verma	1170.00
5	Karan Prasad	7200.00
6	Khushi Pandey	1500.00
7	Manoj Kumar	3400.00
8	Abhishek Singh	6240.00
9	Kamala Yadav	12500.00
10	Tanya Pandey	2670.00

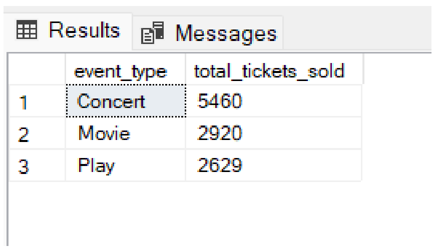
Query executed successfully.

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

SELECT customer_name

FROM Customer c

WHERE EXISTS (SELECT 1

            FROM Booking b

            JOIN Event e ON b.event_id = e.event_id

            WHERE b.customer_id = c.customer_id

            AND e.venue_id = 1);           -- We can replace 1 with any desired venue id
```


Results Messages

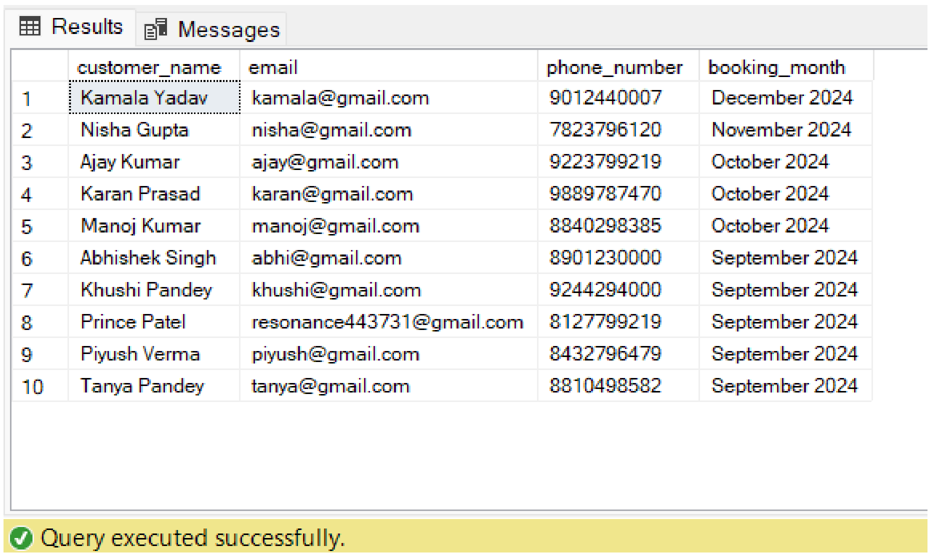
	customer_name
1	Prince Patel

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

SELECT event_type,

      SUM(tickets_sold) AS total_tickets_sold

FROM (
    SELECT event_type,
           (total_seats - available_seats) AS tickets_sold
    FROM Event
) Event
GROUP BY event_type;
```



```
   event_type  total_tickets_sold
1  Concert     5460
2  Movie       2920
3  Play        2629
```

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

SELECT customer_name, email, phone_number,
       (SELECT FORMAT(b.booking_date, 'MMMM yyyy')
        FROM Booking b
        WHERE b.customer_id = c.customer_id
        ) AS booking_month
FROM Customer c
WHERE EXISTS (
    SELECT 1
    FROM Booking b
    WHERE b.customer_id = c.customer_id
)
ORDER BY booking_month;
```

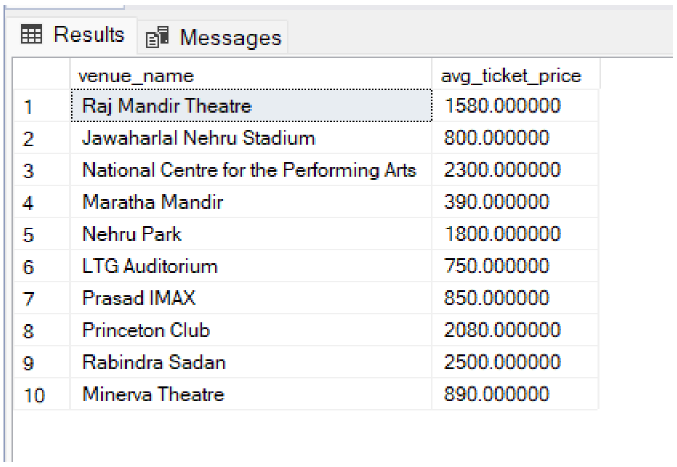


	customer_name	email	phone_number	booking_month
1	Kamala Yadav	kamala@gmail.com	9012440007	December 2024
2	Nisha Gupta	nisha@gmail.com	7823796120	November 2024
3	Ajay Kumar	ajay@gmail.com	9223799219	October 2024
4	Karan Prasad	karan@gmail.com	9889787470	October 2024
5	Manoj Kumar	manoj@gmail.com	8840298385	October 2024
6	Abhishek Singh	abhi@gmail.com	8901230000	September 2024
7	Khushi Pandey	khushi@gmail.com	9244294000	September 2024
8	Prince Patel	resonance443731@gmail.com	8127799219	September 2024
9	Piyush Verma	piyush@gmail.com	8432796479	September 2024
10	Tanya Pandey	tanya@gmail.com	8810498582	September 2024

Query executed successfully.

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

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



	venue_name	avg_ticket_price
1	Raj Mandir Theatre	1580.000000
2	Jawaharlal Nehru Stadium	800.000000
3	National Centre for the Performing Arts	2300.000000
4	Maratha Mandir	390.000000
5	Nehru Park	1800.000000
6	LTG Auditorium	750.000000
7	Prasad IMAX	850.000000
8	Princeton Club	2080.000000
9	Rabindra Sadan	2500.000000
10	Minerva Theatre	890.000000