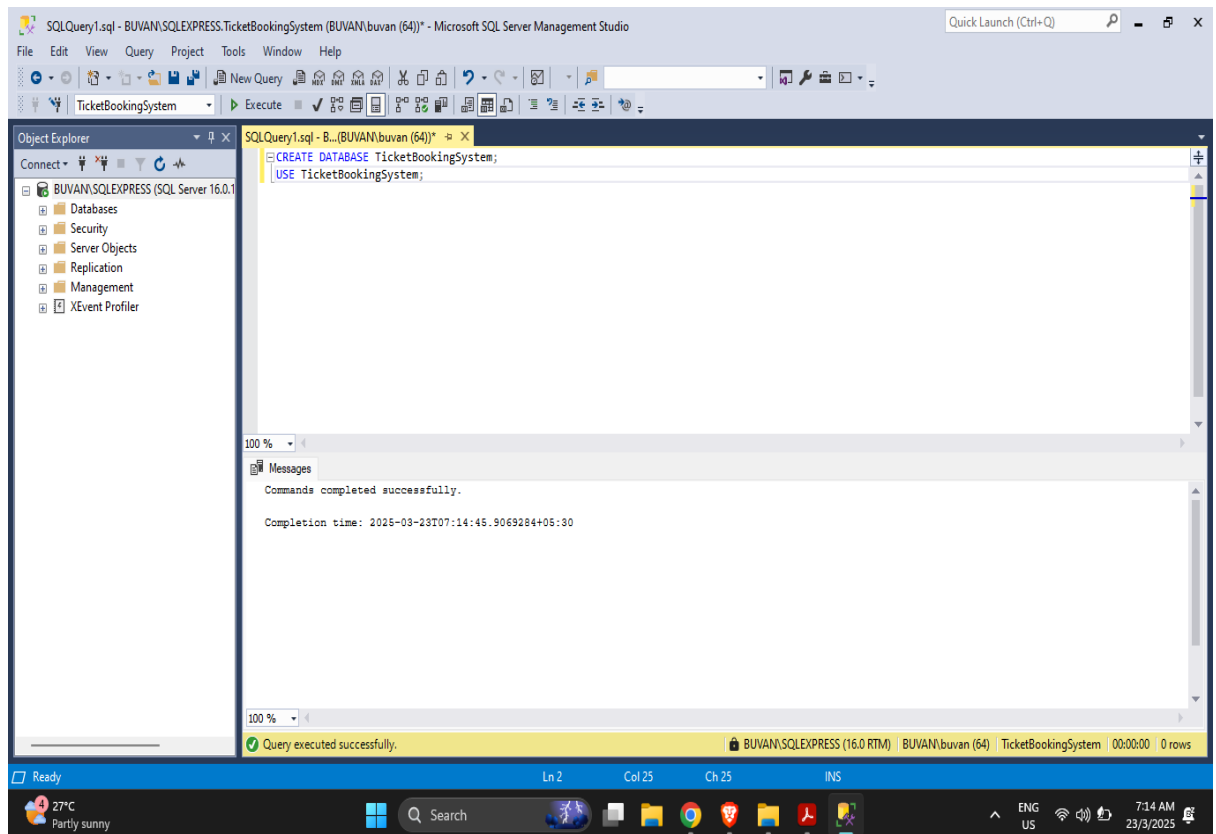


Ticket Booking System

Task 1:

1. Create the database named "TicketBookingSystem"



2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

- Venu
- Event
- Customers
- Booking

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

```

SQLQuery1.sql - B...[BUVAN\buvan (64)]*
--
CREATE TABLE Venue (
    VenueID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100) NOT NULL,
    Location VARCHAR(255) NOT NULL);

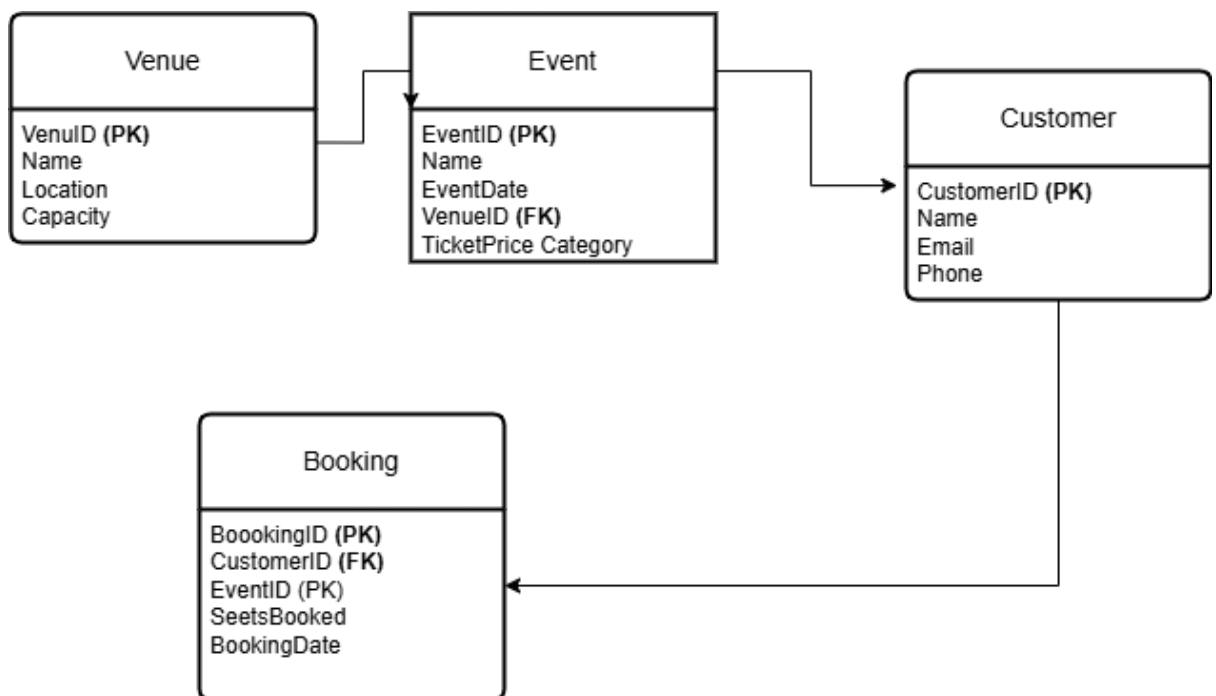
CREATE TABLE Event (
    EventID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100) NOT NULL,
    EventDate DATE NOT NULL,
    EventTime TIME NOT NULL,
    VenueID INT NOT NULL,
    FOREIGN KEY (VenueID) REFERENCES Venue(VenueID) ON DELETE CASCADE
);

CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15) UNIQUE NOT NULL
);

CREATE TABLE Booking (
    BookingID INT PRIMARY KEY IDENTITY(1,1),
    CustomerID INT NOT NULL,
    EventID INT NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE,
    FOREIGN KEY (EventID) REFERENCES Event(EventID) ON DELETE CASCADE
);
--
85 %
Messages
Commands completed successfully.
Completion time: 2025-03-22T07:35:30.4135787+05:30
85 %
Query executed successfully.
BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (64) | TicketBookingSystem | 00:00:00 | 0 rows

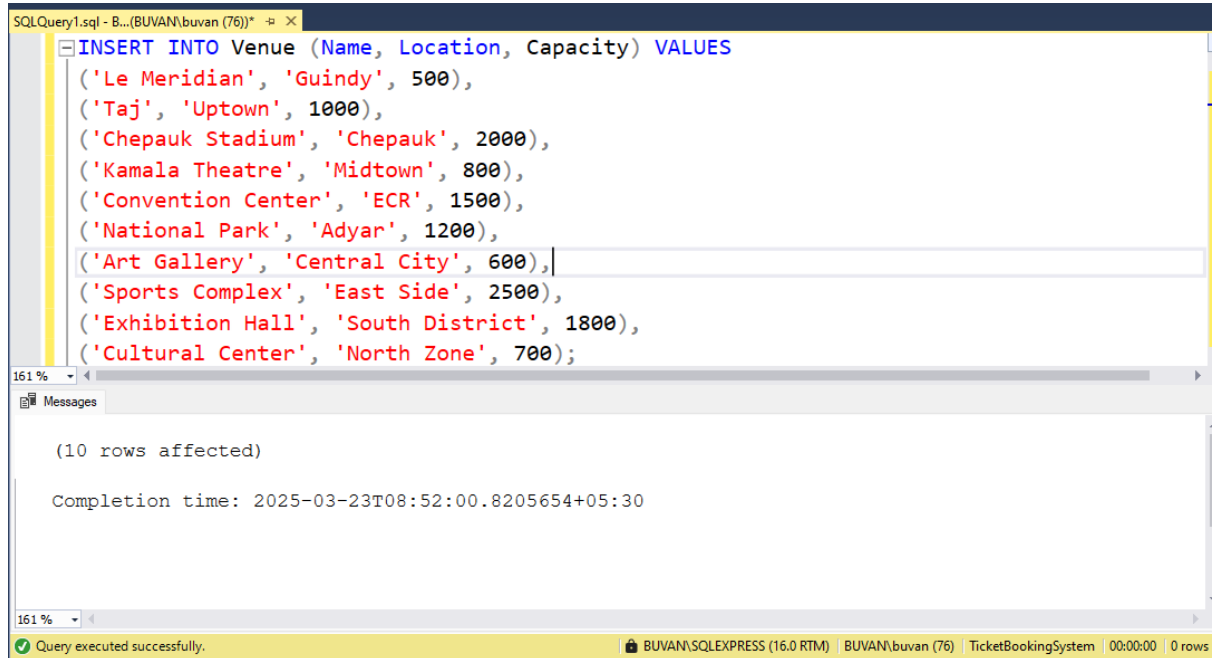
```

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



Task 2:

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



The screenshot shows a SQL query window with the following text:

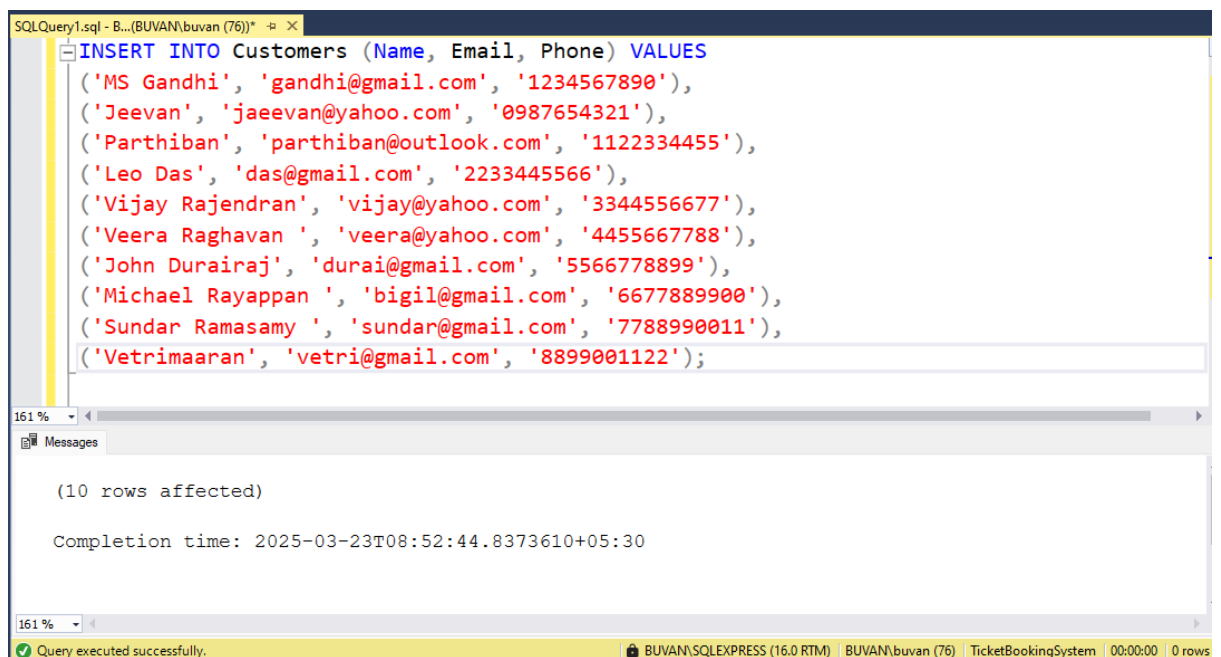
```
INSERT INTO Venue (Name, Location, Capacity) VALUES
('Le Meridian', 'Guindy', 500),
('Taj', 'Uptown', 1000),
('Chepauk Stadium', 'Chepauk', 2000),
('Kamala Theatre', 'Midtown', 800),
('Convention Center', 'ECR', 1500),
('National Park', 'Adyar', 1200),
('Art Gallery', 'Central City', 600),
('Sports Complex', 'East Side', 2500),
('Exhibition Hall', 'South District', 1800),
('Cultural Center', 'North Zone', 700);
```

Below the query, the Messages pane displays:

```
(10 rows affected)

Completion time: 2025-03-23T08:52:00.8205654+05:30
```

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 0 rows



The screenshot shows a SQL query window with the following text:

```
INSERT INTO Customers (Name, Email, Phone) VALUES
('MS Gandhi', 'gandhi@gmail.com', '1234567890'),
('Jeevan', 'jaeevan@yahoo.com', '0987654321'),
('Parthiban', 'parthiban@outlook.com', '1122334455'),
('Leo Das', 'das@gmail.com', '2233445566'),
('Vijay Rajendran', 'vijay@yahoo.com', '3344556677'),
('Veera Raghavan', 'veera@yahoo.com', '4455667788'),
('John Durairaj', 'durai@gmail.com', '5566778899'),
('Michael Rayappan', 'bigil@gmail.com', '6677889900'),
('Sundar Ramasamy', 'sundar@gmail.com', '7788990011'),
('Vetrimaaran', 'vetri@gmail.com', '8899001122');
```

Below the query, the Messages pane displays:

```
(10 rows affected)

Completion time: 2025-03-23T08:52:44.8373610+05:30
```

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 0 rows

```
SQLQuery1.sql - B... (BUVAN\buvan (76)) *  X
INSERT INTO Event (Name, EventDate, EventTime, VenueID) VALUES
('Rock Concert', '2025-06-10', '19:00:00', 1),
('Disco', '2025-07-20', '17:30:00', 3),
('IPL', '2025-08-05', '10:00:00', 5),
('Drama Play', '2025-06-25', '20:00:00', 4),
('Theater Play', '2025-09-10', '18:00:00', 2),
('Music Festival', '2025-07-15', '16:00:00', 7),
('Classical Arts', '2025-08-22', '11:30:00', 6),
('Basketball League', '2025-06-30', '14:00:00', 8),
('Orchestra Night', '2025-07-12', '21:00:00', 9),
('Food Carnival', '2025-08-18', '15:00:00', 10);

(10 rows affected)

Completion time: 2025-03-23T08:53:48.6260019+05:30
```

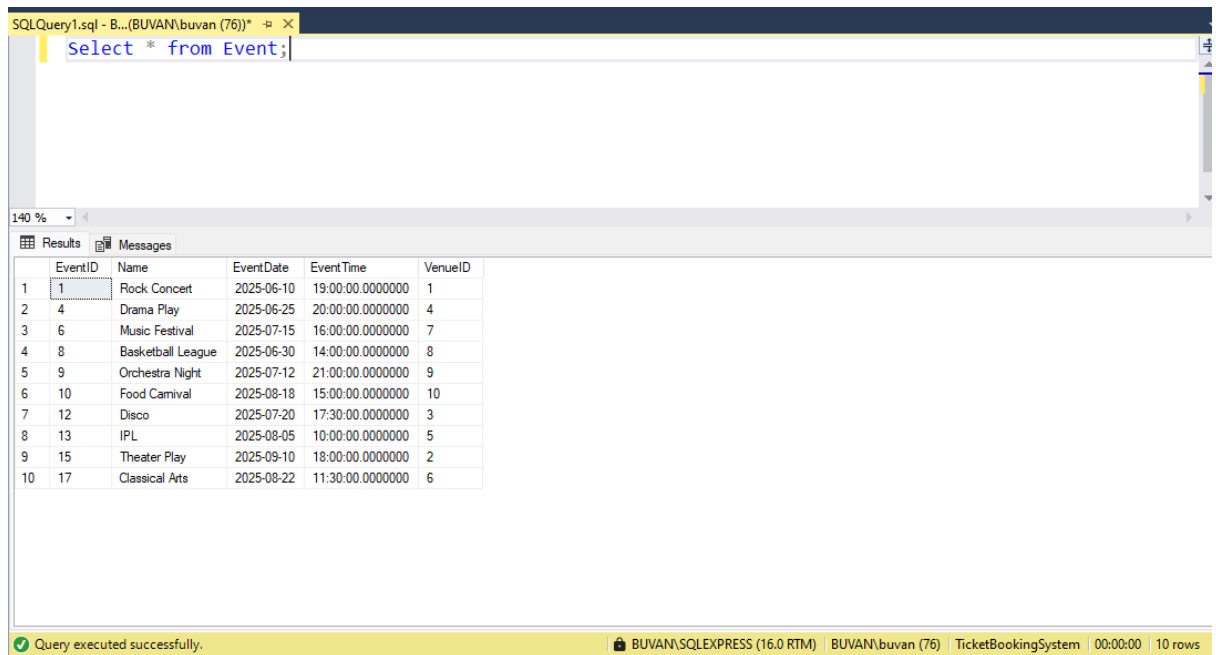
```
SQLQuery1.sql - B... (BUVAN\buvan (76)) *  X
INSERT INTO Booking (CustomerID, EventID, SeatsBooked) VALUES
(1, 1, 2),
(2, 2, 4),
(3, 3, 1),
(4, 4, 3),
(5, 5, 5),
(6, 6, 2),
(7, 7, 6),
(8, 8, 1),
(9, 9, 3),
(10, 10, 7);

(10 rows affected)

Completion time: 2025-03-23T08:54:12.4796558+05:30
```

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 0 rows

2. Write a SQL query to list all Events.



SQLQuery1.sql - B...(BUVAN\buvan (76))*

```
Select * from Event;
```

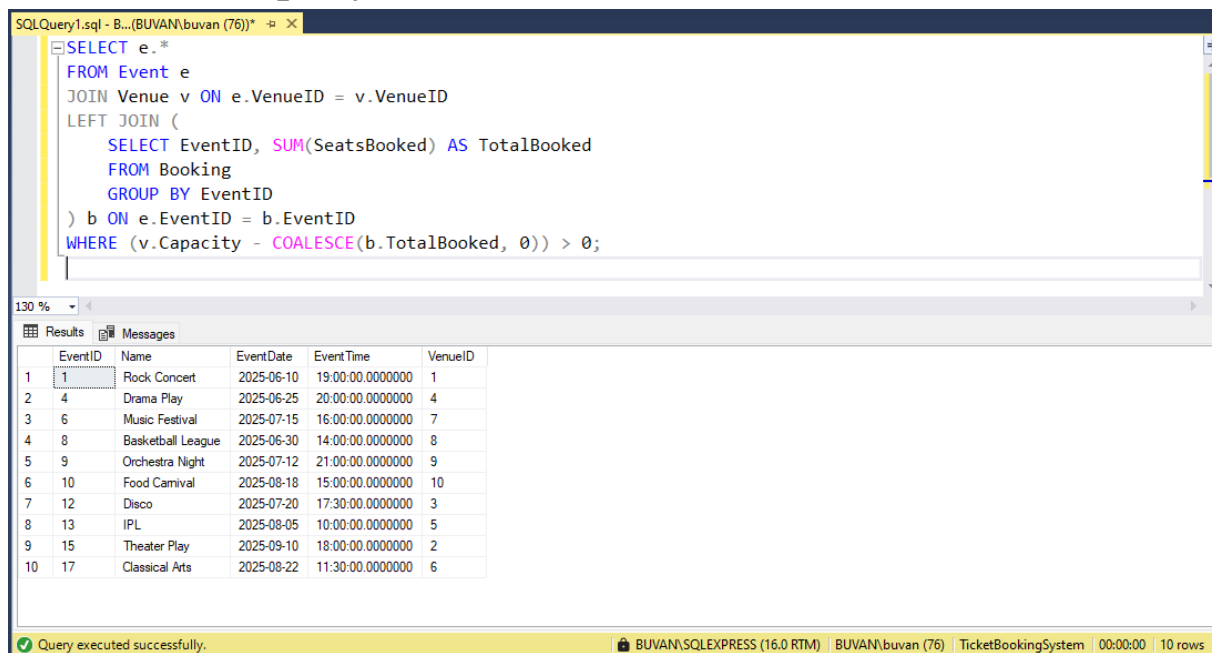
140 %

Results Messages

	EventID	Name	EventDate	EventTime	VenueID
1	1	Rock Concert	2025-06-10	19:00:00.0000000	1
2	4	Drama Play	2025-06-25	20:00:00.0000000	4
3	6	Music Festival	2025-07-15	16:00:00.0000000	7
4	8	Basketball League	2025-06-30	14:00:00.0000000	8
5	9	Orchestra Night	2025-07-12	21:00:00.0000000	9
6	10	Food Carnival	2025-08-18	15:00:00.0000000	10
7	12	Disco	2025-07-20	17:30:00.0000000	3
8	13	IPL	2025-08-05	10:00:00.0000000	5
9	15	Theater Play	2025-09-10	18:00:00.0000000	2
10	17	Classical Arts	2025-08-22	11:30:00.0000000	6

Query executed successfully. BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 10 rows

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



SQLQuery1.sql - B...(BUVAN\buvan (76))*

```
SELECT e.*  
FROM Event e  
JOIN Venue v ON e.VenueID = v.VenueID  
LEFT JOIN (  
    SELECT EventID, SUM(SeatsBooked) AS TotalBooked  
    FROM Booking  
    GROUP BY EventID  
) b ON e.EventID = b.EventID  
WHERE (v.Capacity - COALESCE(b.TotalBooked, 0)) > 0;
```

130 %

Results Messages

	EventID	Name	EventDate	EventTime	VenueID
1	1	Rock Concert	2025-06-10	19:00:00.0000000	1
2	4	Drama Play	2025-06-25	20:00:00.0000000	4
3	6	Music Festival	2025-07-15	16:00:00.0000000	7
4	8	Basketball League	2025-06-30	14:00:00.0000000	8
5	9	Orchestra Night	2025-07-12	21:00:00.0000000	9
6	10	Food Carnival	2025-08-18	15:00:00.0000000	10
7	12	Disco	2025-07-20	17:30:00.0000000	3
8	13	IPL	2025-08-05	10:00:00.0000000	5
9	15	Theater Play	2025-09-10	18:00:00.0000000	2
10	17	Classical Arts	2025-08-22	11:30:00.0000000	6

Query executed successfully. BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 10 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (76))

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

```
SELECT * FROM Event WHERE Name LIKE '%play%';
```

130 %

Results Messages

	EventID	Name	EventDate	EventTime	VenueID
1	4	Drama Play	2025-06-25	20:00:00.0000000	4
2	15	Theater Play	2025-09-10	18:00:00.0000000	2

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 2 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (76))

```
SELECT * FROM Event WHERE TicketPrice BETWEEN 1000 AND 2500;
```

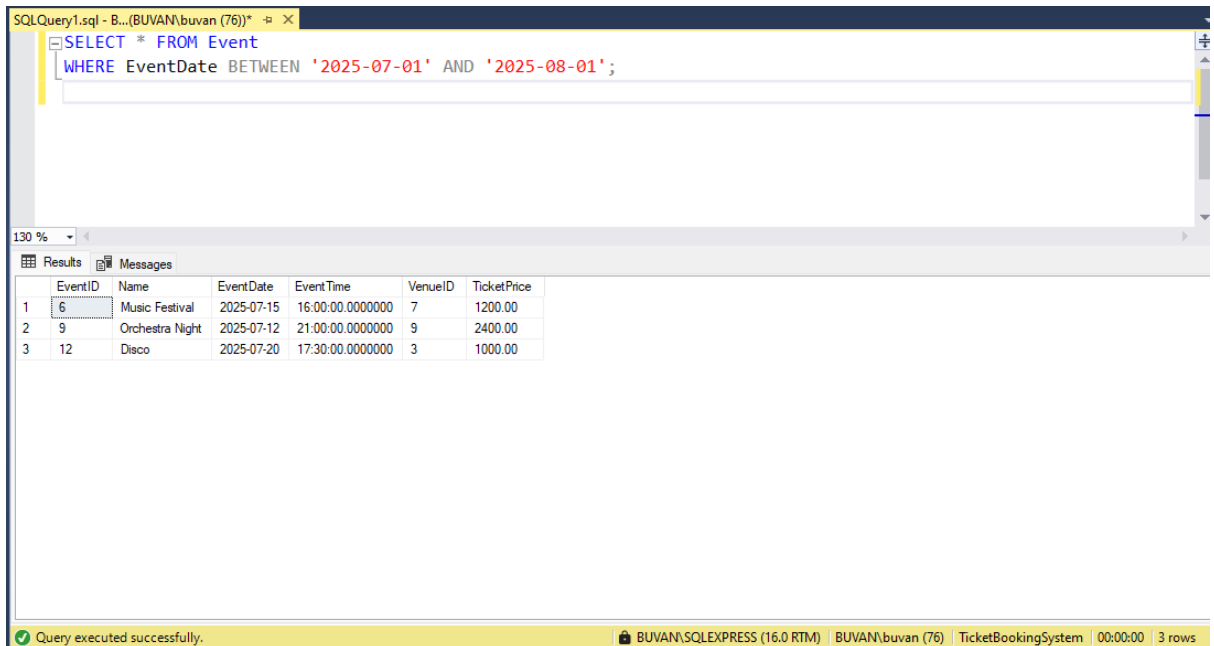
130 %

Results Messages

	EventID	Name	EventDate	EventTime	VenueID	TicketPrice
1	1	Rock Concert	2025-06-10	19:00:00.0000000	1	2000.00
2	6	Music Festival	2025-07-15	16:00:00.0000000	7	1200.00
3	8	Basketball League	2025-06-30	14:00:00.0000000	8	2200.00
4	9	Orchestra Night	2025-07-12	21:00:00.0000000	9	2400.00
5	10	Food Carnival	2025-08-18	15:00:00.0000000	10	1500.00
6	12	Disco	2025-07-20	17:30:00.0000000	3	1000.00
7	13	IPL	2025-08-05	10:00:00.0000000	5	1000.00
8	15	Theater Play	2025-09-10	18:00:00.0000000	2	1000.00
9	17	Classical Arts	2025-08-22	11:30:00.0000000	6	1000.00

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 9 rows

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



The screenshot shows a SQL query window with the following query:

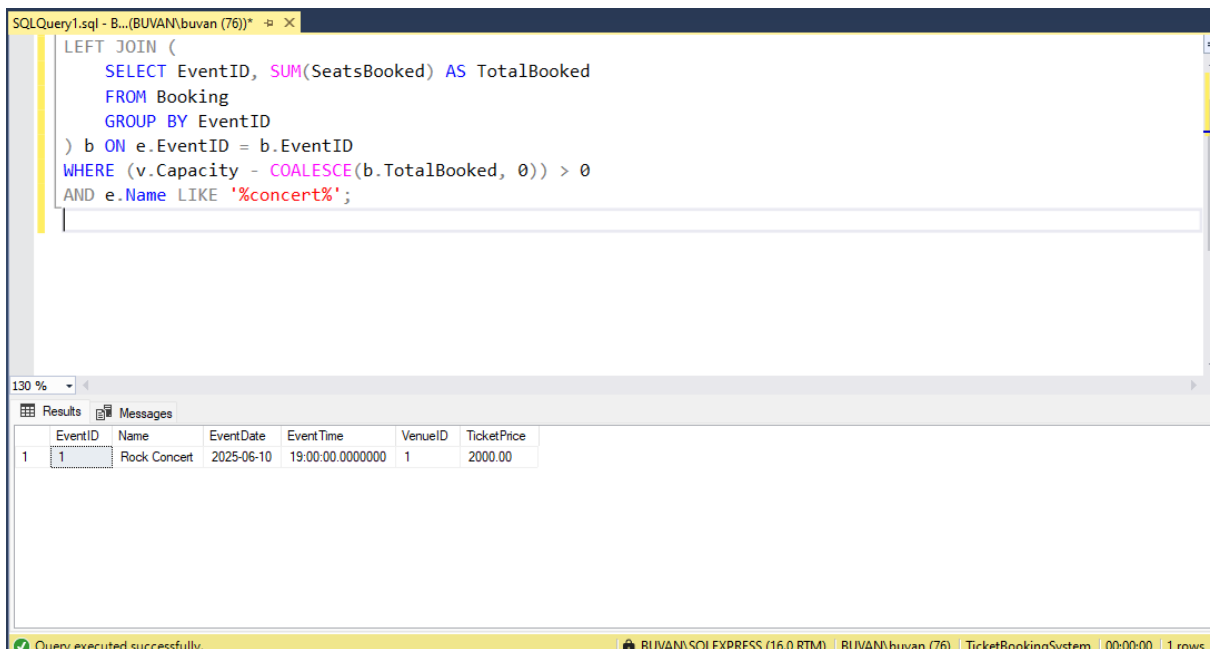
```
SELECT * FROM Event
WHERE EventDate BETWEEN '2025-07-01' AND '2025-08-01';
```

The results pane displays the following data:

	EventID	Name	EventDate	EventTime	VenueID	TicketPrice
1	6	Music Festival	2025-07-15	16:00:00.0000000	7	1200.00
2	9	Orchestra Night	2025-07-12	21:00:00.0000000	9	2400.00
3	12	Disco	2025-07-20	17:30:00.0000000	3	1000.00

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 3 rows

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



The screenshot shows a SQL query window with the following query:

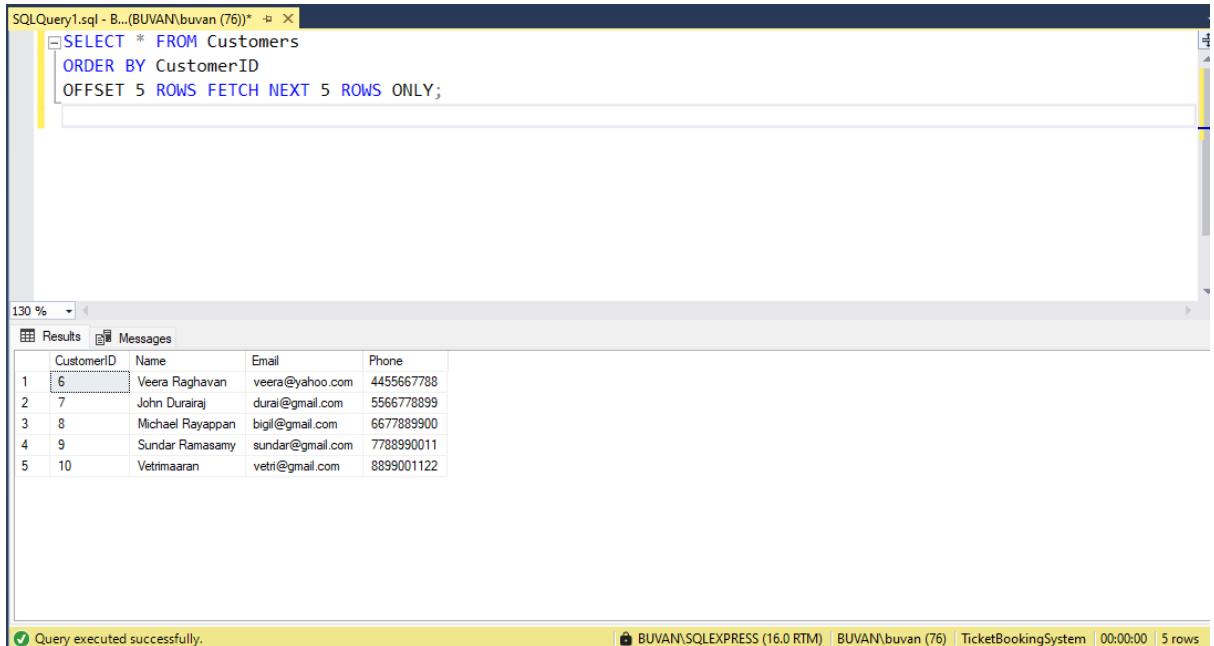
```
LEFT JOIN (
    SELECT EventID, SUM(SeatsBooked) AS TotalBooked
    FROM Booking
    GROUP BY EventID
) b ON e.EventID = b.EventID
WHERE (v.Capacity - COALESCE(b.TotalBooked, 0)) > 0
AND e.Name LIKE '%concert%';
```

The results pane displays the following data:

	EventID	Name	EventDate	EventTime	VenueID	TicketPrice
1	1	Rock Concert	2025-06-10	19:00:00.0000000	1	2000.00

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 1 rows

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



The screenshot shows a SQL query window with the following query:

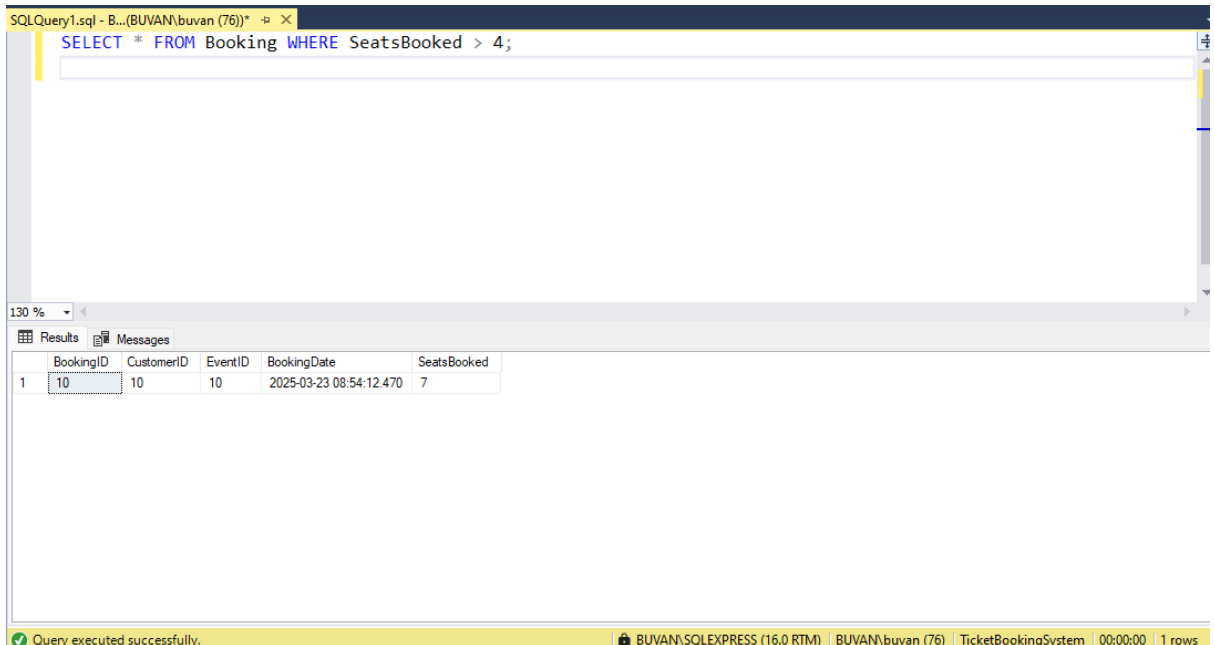
```
SELECT * FROM Customers
ORDER BY CustomerID
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

The results pane displays a table with 5 rows and 4 columns: CustomerID, Name, Email, and Phone.

	CustomerID	Name	Email	Phone
1	6	Veera Raghavan	veera@yahoo.com	4455667788
2	7	John Durairaj	durai@gmail.com	5566778899
3	8	Michael Rayappan	bigil@gmail.com	6677889900
4	9	Sundar Ramasamy	sundar@gmail.com	7788990011
5	10	Vetrimaaran	vetri@gmail.com	8899001122

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 5 rows

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



The screenshot shows a SQL query window with the following query:

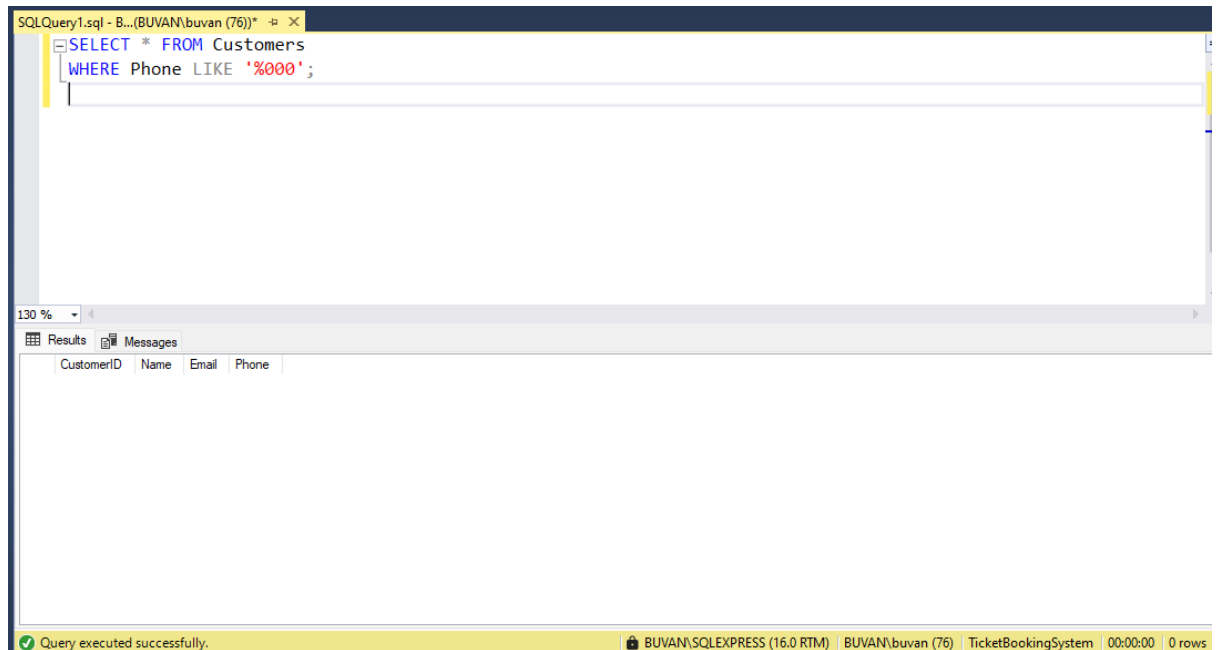
```
SELECT * FROM Booking WHERE SeatsBooked > 4;
```

The results pane displays a table with 1 row and 6 columns: BookingID, CustomerID, EventID, BookingDate, and SeatsBooked.

	BookingID	CustomerID	EventID	BookingDate	SeatsBooked
1	10	10	10	2025-03-23 08:54:12.470	7

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 1 rows

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

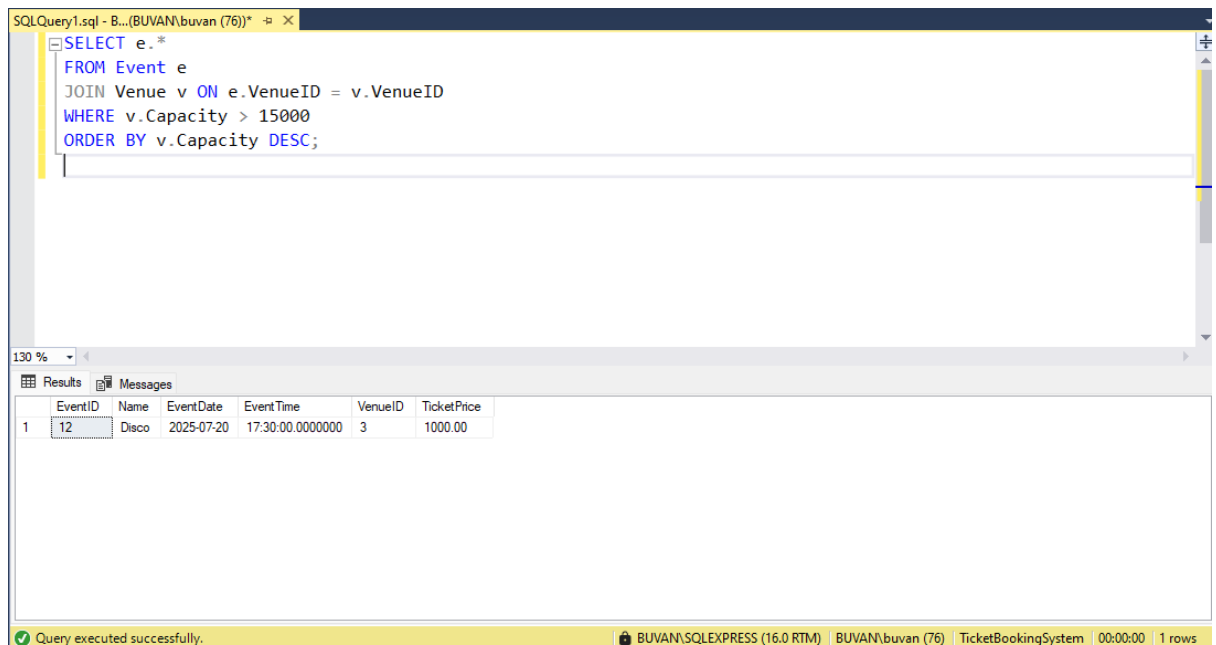


The screenshot shows the SQL Server Enterprise Manager interface. The query editor contains the following SQL query:

```
SELECT * FROM Customers
WHERE Phone LIKE '%000';
```

The query was executed successfully, as indicated by the status bar at the bottom: "Query executed successfully. BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 0 rows". The Results pane is empty, showing only the column headers: CustomerID, Name, Email, and Phone.

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



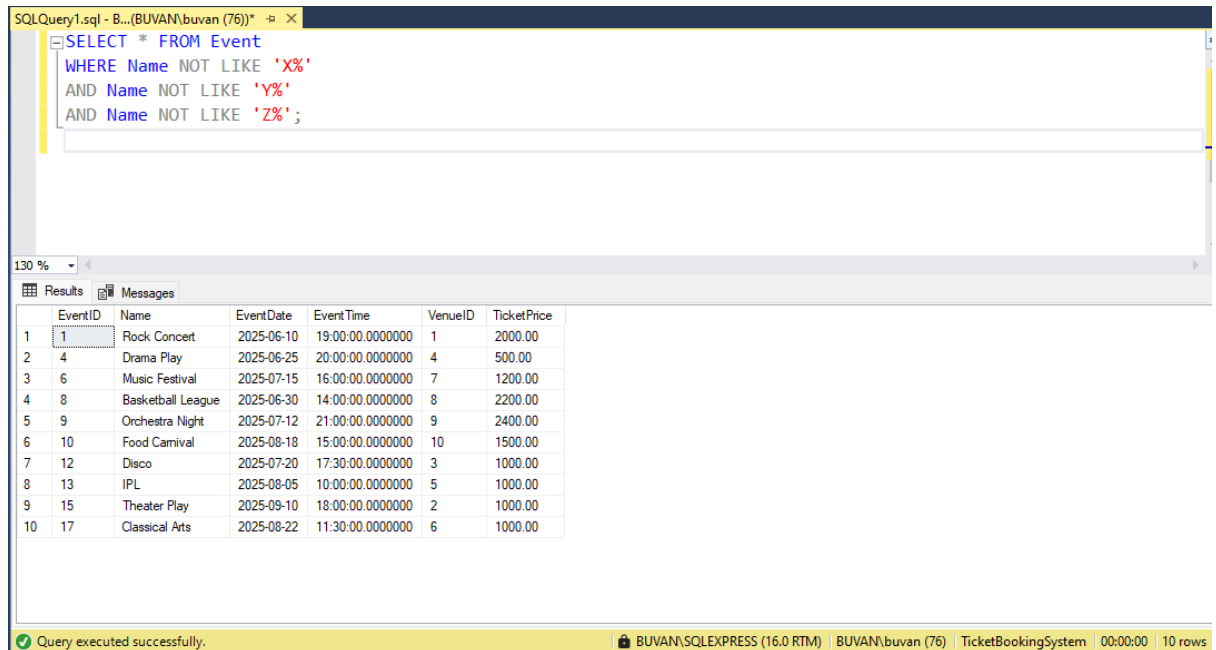
The screenshot shows the SQL Server Enterprise Manager interface. The query editor contains the following SQL query:

```
SELECT e.*
FROM Event e
JOIN Venue v ON e.VenueID = v.VenueID
WHERE v.Capacity > 15000
ORDER BY v.Capacity DESC;
```

The query was executed successfully, as indicated by the status bar at the bottom: "Query executed successfully. BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 1 rows". The Results pane displays one row of data:

EventID	Name	EventDate	EventTime	VenueID	TicketPrice
12	Disco	2025-07-20	17:30:00.0000000	3	1000.00

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



The screenshot shows a SQL query window with the following query:

```
SELECT * FROM Event
WHERE Name NOT LIKE 'X%'
AND Name NOT LIKE 'Y%'
AND Name NOT LIKE 'Z%';
```

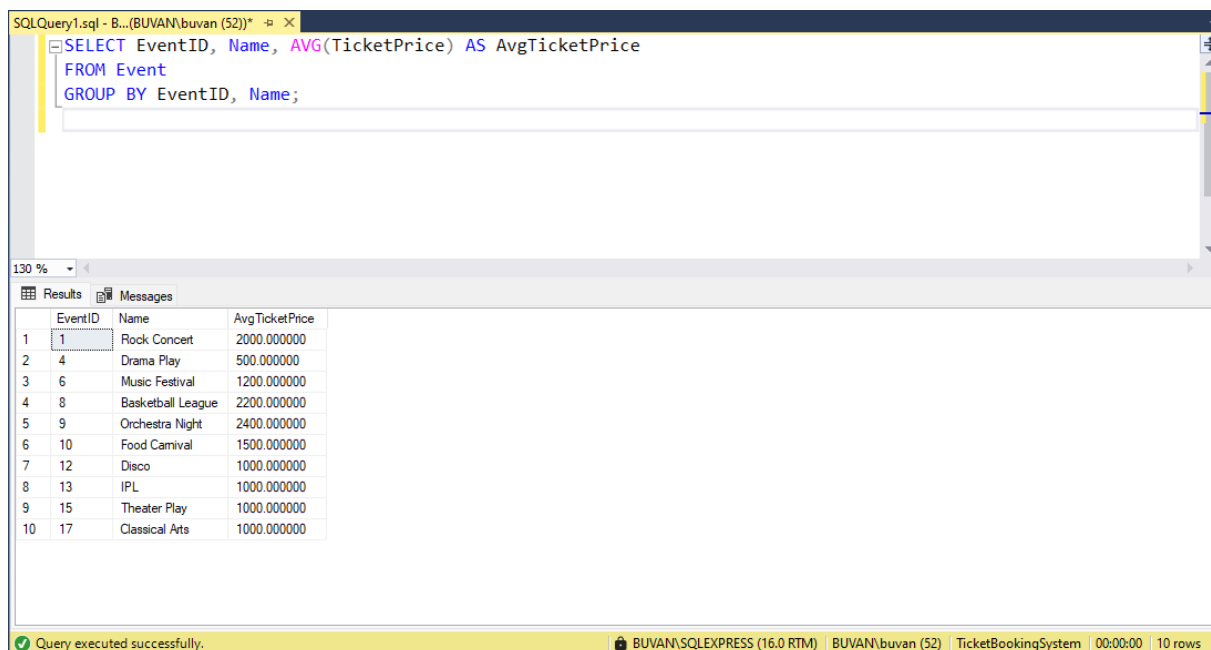
The results pane displays the following data:

EventID	Name	EventDate	EventTime	VenueID	TicketPrice
1	Rock Concert	2025-06-10	19:00:00.0000000	1	2000.00
2	Drama Play	2025-06-25	20:00:00.0000000	4	500.00
3	Music Festival	2025-07-15	16:00:00.0000000	7	1200.00
4	Basketball League	2025-06-30	14:00:00.0000000	8	2200.00
5	Orchestra Night	2025-07-12	21:00:00.0000000	9	2400.00
6	Food Carnival	2025-08-18	15:00:00.0000000	10	1500.00
7	Disco	2025-07-20	17:30:00.0000000	3	1000.00
8	IPL	2025-08-05	10:00:00.0000000	5	1000.00
9	Theater Play	2025-09-10	18:00:00.0000000	2	1000.00
10	Classical Arts	2025-08-22	11:30:00.0000000	6	1000.00

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (76) | TicketBookingSystem | 00:00:00 | 10 rows

Task 3:

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



The screenshot shows a SQL query window with the following query:

```
SELECT EventID, Name, AVG(TicketPrice) AS AvgTicketPrice
FROM Event
GROUP BY EventID, Name;
```

The results pane displays the following data:

EventID	Name	AvgTicketPrice
1	Rock Concert	2000.000000
2	Drama Play	500.000000
3	Music Festival	1200.000000
4	Basketball League	2200.000000
5	Orchestra Night	2400.000000
6	Food Carnival	1500.000000
7	Disco	1000.000000
8	IPL	1000.000000
9	Theater Play	1000.000000
10	Classical Arts	1000.000000

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 10 rows

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

The screenshot shows a SQL query window with the following query:

```
SELECT e.EventID, e.Name, SUM(b.SeatsBooked * e.TicketPrice) AS TotalRevenue
FROM Event e
JOIN Booking b ON e.EventID = b.EventID
GROUP BY e.EventID, e.Name;
```

The query results are displayed in a table with 6 rows:

EventID	Name	TotalRevenue
1	Rock Concert	4000.00
2	Drama Play	1500.00
3	Music Festival	2400.00
4	Basketball League	2200.00
5	Orchestra Night	7200.00
6	Food Carnival	10500.00

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 6 rows

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

The screenshot shows a SQL query window with the following query:

```
SELECT TOP 1 e.EventID, e.Name, SUM(b.SeatsBooked) AS TotalTicketsSold
FROM Event e
JOIN Booking b ON e.EventID = b.EventID
GROUP BY e.EventID, e.Name
ORDER BY TotalTicketsSold DESC;
```

The query results are displayed in a table with 1 row:

EventID	Name	TotalTicketsSold
10	Food Carnival	7

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 1 rows

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

The screenshot shows a SQL query window with the following query:

```
SELECT e.EventID, e.Name, SUM(b.SeatsBooked) AS TotalTicketsSold
FROM Event e
JOIN Booking b ON e.EventID = b.EventID
GROUP BY e.EventID, e.Name;
```

The query results are displayed in a table with 6 rows:

EventID	Name	TotalTicketsSold
1	Rock Concert	2
2	Drama Play	3
3	Music Festival	2
4	Basketball League	1
5	Orchestra Night	3
6	Food Carnival	7

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 6 rows

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

The screenshot shows a SQL query window with the following query:

```
SELECT e.EventID, e.Name
FROM Event e
LEFT JOIN Booking b ON e.EventID = b.EventID
WHERE b.EventID IS NULL;
```

The query results are displayed in a table with 4 rows:

EventID	Name
12	Disco
13	IPL
15	Theater Play
17	Classical Arts

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 4 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (52))*

```

SELECT TOP 1 c.CustomerID, c.Name, SUM(b.SeatsBooked) AS TotalTickets
FROM Customer c
JOIN Booking b ON c.CustomerID = b.CustomerID
GROUP BY c.CustomerID, c.Name
ORDER BY TotalTickets DESC;

```

Results

	CustomerID	Name	TotalTickets
1	10	Vetrimaaran	7

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 1 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (52))*

```

SELECT MONTH(e.EventDate) AS EventMonth, e.Name, SUM(b.SeatsBooked) AS TotalTickets
FROM Event e
JOIN Booking b ON e.EventID = b.EventID
GROUP BY MONTH(e.EventDate), e.Name
ORDER BY EventMonth;

```

Results

	EventMonth	Name	TotalTickets
1	6	Basketball League	1
2	6	Drama Play	3
3	6	Rock Concert	2
4	7	Music Festival	2
5	7	Orchestra Night	3
6	8	Food Carnival	7

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 6 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (52))

```

SELECT v.VenueID, v.Name AS VenueName, AVG(e.TicketPrice) AS AvgTicketPrice
FROM Venue v
JOIN Event e ON v.VenueID = e.VenueID
GROUP BY v.VenueID, v.Name;

```

130 %

	VenueID	VenueName	AvgTicketPrice
1	1	Le Meridian	2000.000000
2	2	Taj	1000.000000
3	3	Chepauk Stadium	1000.000000
4	4	Kamala Theatre	500.000000
5	5	Convention Center	1000.000000
6	6	National Park	1000.000000
7	7	Art Gallery	1200.000000
8	8	Sports Complex	2200.000000
9	9	Exhibition Hall	2400.000000
10	10	Cultural Center	1500.000000

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 10 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (52))

```

SELECT e.EventType, SUM(b.SeatsBooked) AS TotalTicketsSold
FROM Event e
JOIN Booking b ON e.EventID = b.EventID
GROUP BY e.EventType;

```

130 %

	Event Type	TotalTicketsSold
1	Concert	2
2	Other	13
3	Theater	3

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 3 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (52))

```

SELECT YEAR(e.EventDate) AS EventYear,
       SUM(b.SeatsBooked * e.TicketPrice) AS TotalRevenue
FROM Event e
JOIN Booking b ON e.EventID = b.EventID
GROUP BY YEAR(e.EventDate)
ORDER BY EventYear;

```

130 %

Results Messages

	EventYear	TotalRevenue
1	2025	27800.00

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 1 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (52))

```

SELECT b.CustomerID, c.Name, COUNT(DISTINCT b.EventID) AS TotalEventsBooked
FROM Booking b
JOIN Customers c ON b.CustomerID = c.CustomerID
GROUP BY b.CustomerID, c.Name
HAVING COUNT(DISTINCT b.EventID) > 1;

```

130 %

Results Messages

	CustomerID	Name	TotalEventsBooked
--	------------	------	-------------------

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (52) | TicketBookingSystem | 00:00:00 | 0 rows

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

SQLQuery1.sql - B...(BUVAN\buvan (78))*

```

SELECT b.CustomerID, c.Name, SUM(b.SeatsBooked) AS TotalTickets
FROM Booking b
JOIN Customers c ON b.CustomerID = c.CustomerID
WHERE b.BookingDate >= DATEADD(DAY, -30, GETDATE())
GROUP BY b.CustomerID, c.Name;

```

130 %

Results Messages

	CustomerID	Name	TotalTickets
1	1	MS Gandhi	2
2	4	Leo Das	3
3	6	Veera Raghavan	2
4	8	Michael Rayappan	1
5	9	Sundar Ramasamy	3
6	10	Vetrimaaran	7

Query executed successfully.

BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 6 rows

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

SQLQuery1.sql - B...(BUVAN\buvan (78))*

```

SELECT e.Category, v.Name AS VenueName, AVG(e.TicketPrice) AS AvgTicketPrice
FROM Event e
JOIN Venue v ON e.VenueID = v.VenueID
GROUP BY e.Category, v.Name;

```

130 %

Results Messages

	Category	VenueName	AvgTicketPrice
1	Other	Art Gallery	1200.000000
2	Other	Chepauk Stadium	1000.000000
3	Other	Convention Center	1000.000000
4	Other	Cultural Center	1500.000000
5	Other	Exhibition Hall	2400.000000
6	Theater	Kamala Theatre	500.000000
7	Concert	Le Meridian	2000.000000
8	Other	National Park	1000.000000
9	Sports	Sports Complex	2200.000000
10	Theater	Taj	1000.000000

Query executed successfully.

BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 10 rows

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

SQLQuery1.sql - B... (BUVAN\buvan (78))*

```

SELECT b.CustomerID, c.Name, SUM(b.SeatsBooked * e.TicketPrice) AS TotalRevenue
FROM Booking b
JOIN Customers c ON b.CustomerID = c.CustomerID
JOIN Event e ON b.EventID = e.EventID
GROUP BY b.CustomerID, c.Name;

```

130 %

Results Messages

	CustomerID	Name	TotalRevenue
1	1	MS Gandhi	4000.00
2	4	Leo Das	1500.00
3	6	Veera Raghavan	2400.00
4	8	Michael Rayappan	2200.00
5	9	Sundar Ramasamy	7200.00
6	10	Vetrimaaran	10500.00

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 6 rows

Task 4:

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

SQLQuery1.sql - B... (BUVAN\buvan (78))*

```

SELECT VenueID, Name,
       (SELECT AVG(TicketPrice)
        FROM Event e
        WHERE e.VenueID = v.VenueID) AS AvgTicketPrice
FROM Venue v;

```

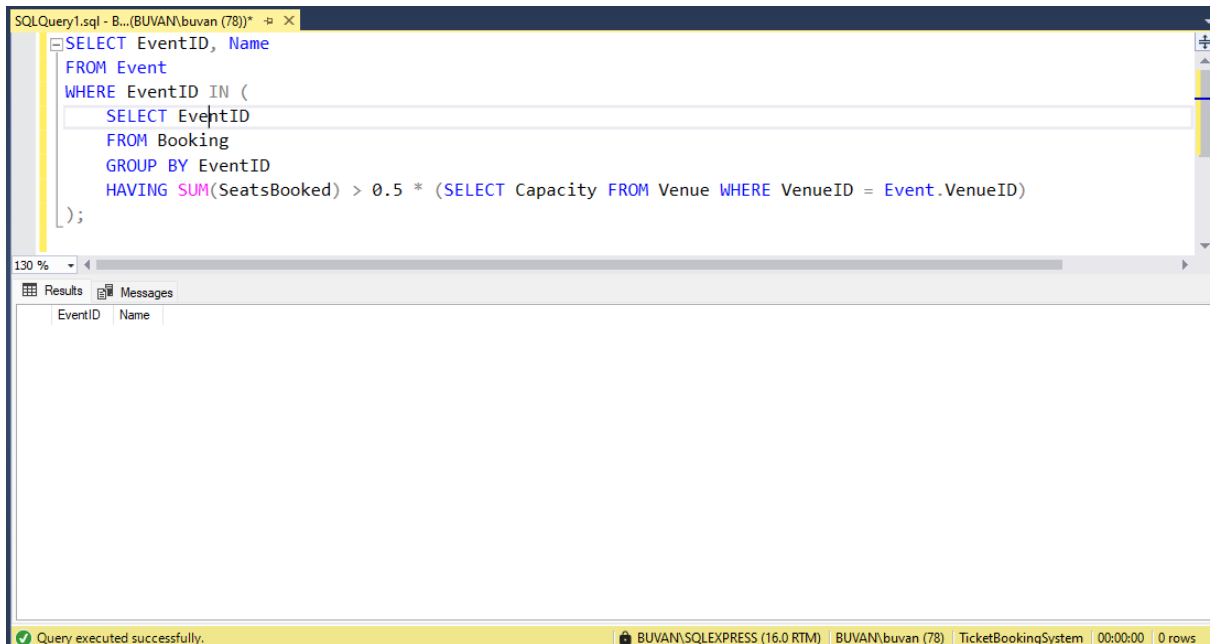
130 %

Results Messages

	VenueID	Name	AvgTicketPrice
1	1	Le Meridian	2000.000000
2	2	Taj	1000.000000
3	3	Chepauk Stadium	1000.000000
4	4	Kamala Theatre	500.000000
5	5	Convention Center	1000.000000
6	6	National Park	1000.000000
7	7	Art Gallery	1200.000000
8	8	Sports Complex	2200.000000
9	9	Exhibition Hall	2400.000000
10	10	Cultural Center	1500.000000

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 10 rows

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



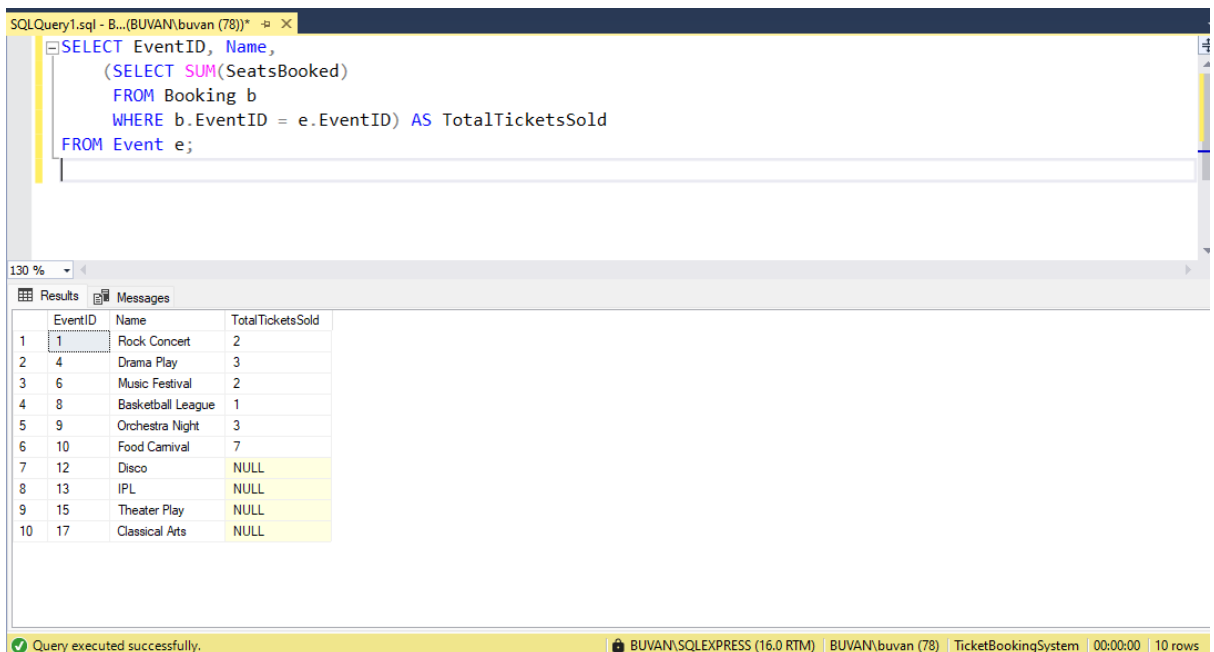
```
SQLQuery1.sql - B... (BUVAN\buvan (78)) * - X
SELECT EventID, Name
FROM Event
WHERE EventID IN (
    SELECT EventID
    FROM Booking
    GROUP BY EventID
    HAVING SUM(SeatsBooked) > 0.5 * (SELECT Capacity FROM Venue WHERE VenueID = Event.VenueID)
);
```

Results Messages

EventID	Name
---------	------

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 0 rows

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



```
SQLQuery1.sql - B... (BUVAN\buvan (78)) * - X
SELECT EventID, Name,
    (SELECT SUM(SeatsBooked)
     FROM Booking b
     WHERE b.EventID = e.EventID) AS TotalTicketsSold
FROM Event e;
```

Results Messages

EventID	Name	TotalTicketsSold
1	Rock Concert	2
2	Drama Play	3
3	Music Festival	2
4	Basketball League	1
5	Orchestra Night	3
6	Food Carnival	7
7	Disco	NULL
8	IPL	NULL
9	Theater Play	NULL
10	Classical Arts	NULL

Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 10 rows

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

The screenshot shows a SQL query window with the following text:

```
SELECT CustomerID, Name
FROM Customers c
WHERE NOT EXISTS (
    SELECT 1
    FROM Booking b
    WHERE b.CustomerID = c.CustomerID
);
```

Below the query window, the Results pane displays a table with 4 rows:

	CustomerID	Name
1	2	Jeevan
2	3	Parthiban
3	5	Vijay Rajendran
4	7	John Durairaj

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 4 rows

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

The screenshot shows a SQL query window with the following text:

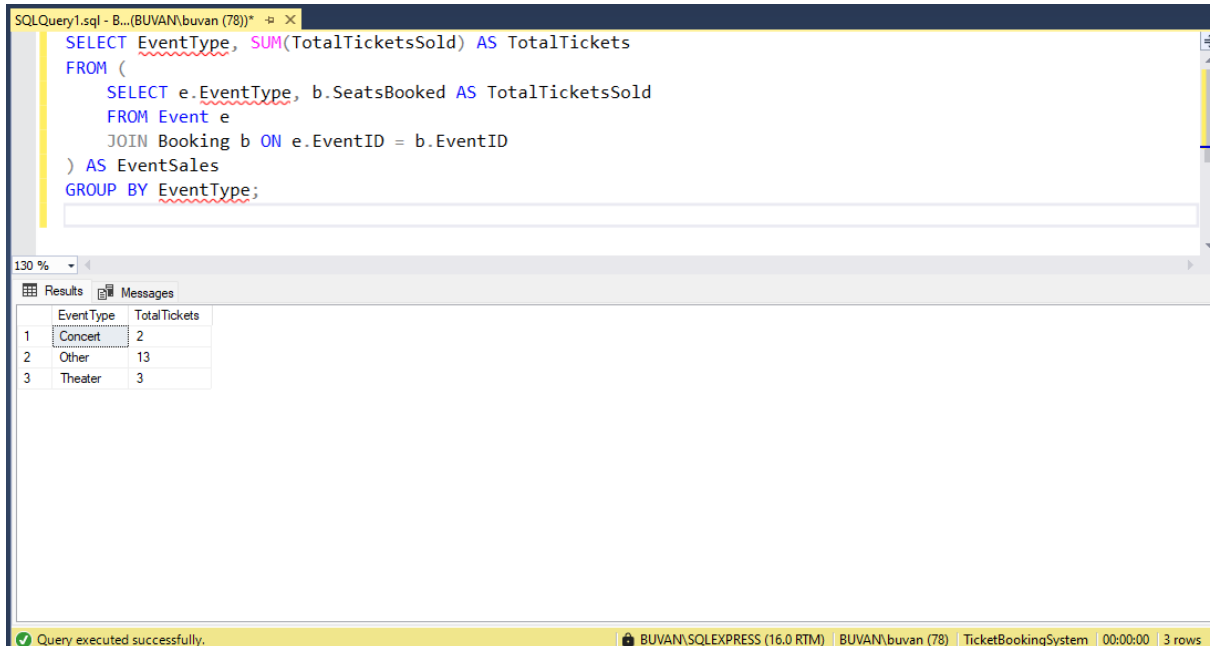
```
SELECT EventID, Name
FROM Event
WHERE EventID NOT IN (
    SELECT DISTINCT EventID FROM Booking
);
```

Below the query window, the Results pane displays a table with 4 rows:

	EventID	Name
1	12	Disco
2	13	IPL
3	15	Theater Play
4	17	Classical Arts

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 4 rows

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



The screenshot shows a SQL query window with the following query:

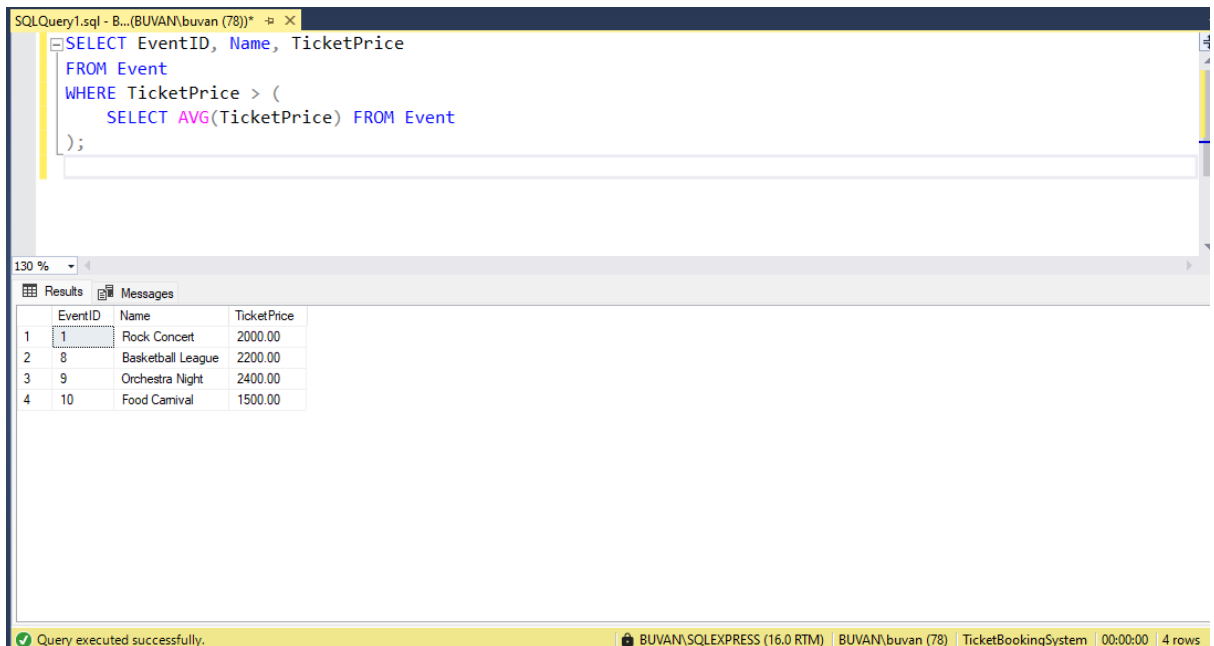
```
SELECT EventType, SUM(TotalTicketsSold) AS TotalTickets
FROM (
    SELECT e.EventType, b.SeatsBooked AS TotalTicketsSold
    FROM Event e
    JOIN Booking b ON e.EventID = b.EventID
) AS EventSales
GROUP BY EventType;
```

The query results are displayed in a table with two columns: Event Type and Total Tickets.

Event Type	Total Tickets
Concert	2
Other	13
Theater	3

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 3 rows

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



The screenshot shows a SQL query window with the following query:

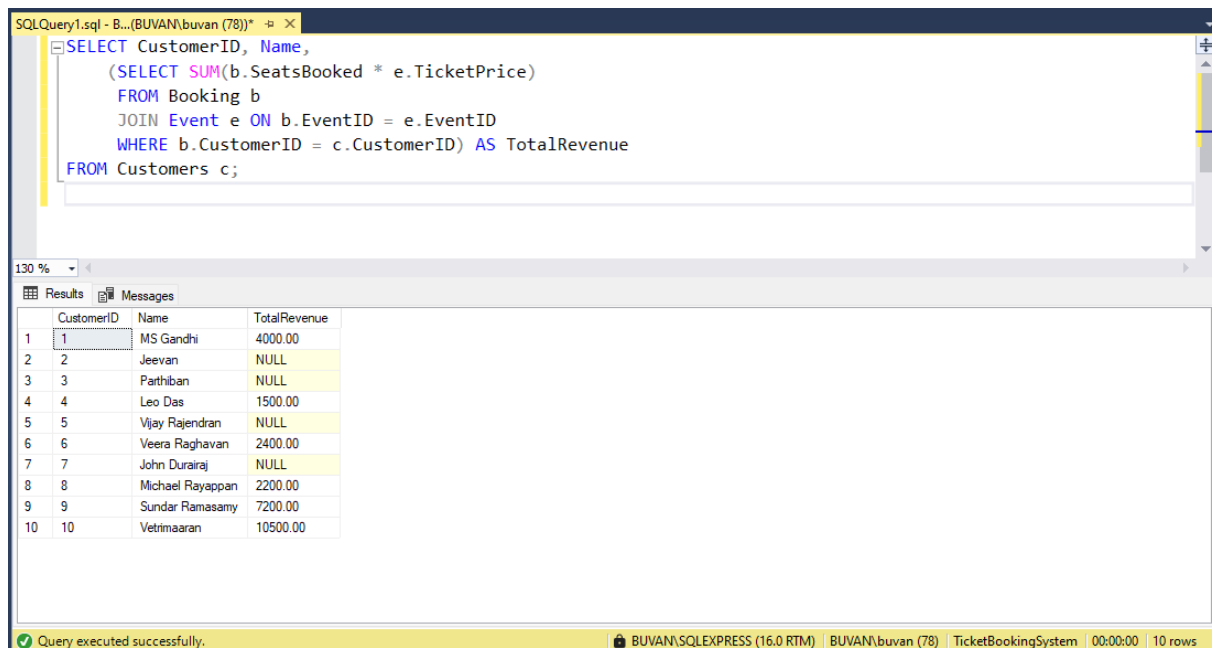
```
SELECT EventID, Name, TicketPrice
FROM Event
WHERE TicketPrice > (
    SELECT AVG(TicketPrice) FROM Event
);
```

The query results are displayed in a table with three columns: EventID, Name, and TicketPrice.

EventID	Name	TicketPrice
1	Rock Concert	2000.00
8	Basketball League	2200.00
9	Orchestra Night	2400.00
10	Food Carnival	1500.00

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 4 rows

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



The screenshot shows a SQL query window with the following query:

```
SELECT CustomerID, Name,  
       (SELECT SUM(b.SeatsBooked * e.TicketPrice)  
        FROM Booking b  
        JOIN Event e ON b.EventID = e.EventID  
        WHERE b.CustomerID = c.CustomerID) AS TotalRevenue  
FROM Customers c;
```

The query results are displayed in a table with 10 rows:

CustomerID	Name	TotalRevenue
1	MS Gandhi	4000.00
2	Jeevan	NULL
3	Parthiban	NULL
4	Leo Das	1500.00
5	Vijay Rajendran	NULL
6	Veera Raghavan	2400.00
7	John Durairaj	NULL
8	Michael Rayappan	2200.00
9	Sundar Ramasamy	7200.00
10	Vetrimaaran	10500.00

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 10 rows

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



The screenshot shows a SQL query window with the following query:

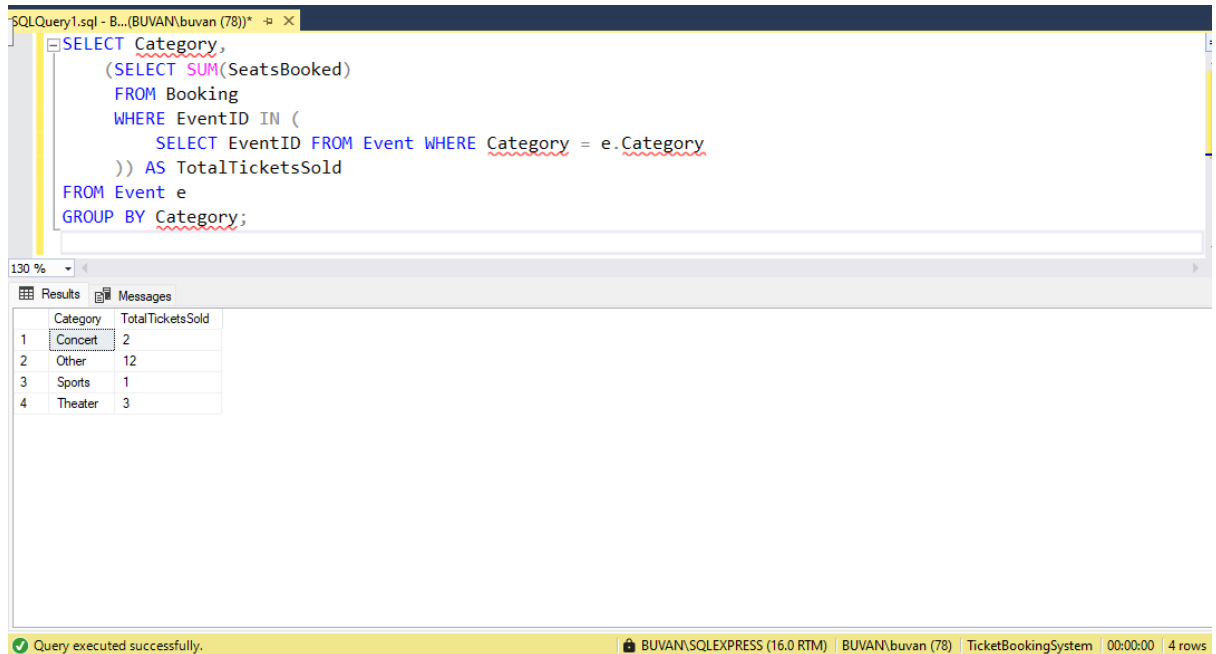
```
FROM Customers  
WHERE CustomerID IN (  
    SELECT DISTINCT CustomerID  
    FROM Booking  
    WHERE EventID IN (  
        SELECT EventID FROM Event WHERE VenueID = 2  
    )  
);
```

The query results are displayed in a table with 2 columns:

CustomerID	Name
------------	------

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 0 rows

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



The screenshot shows a SQL query window with the following query:

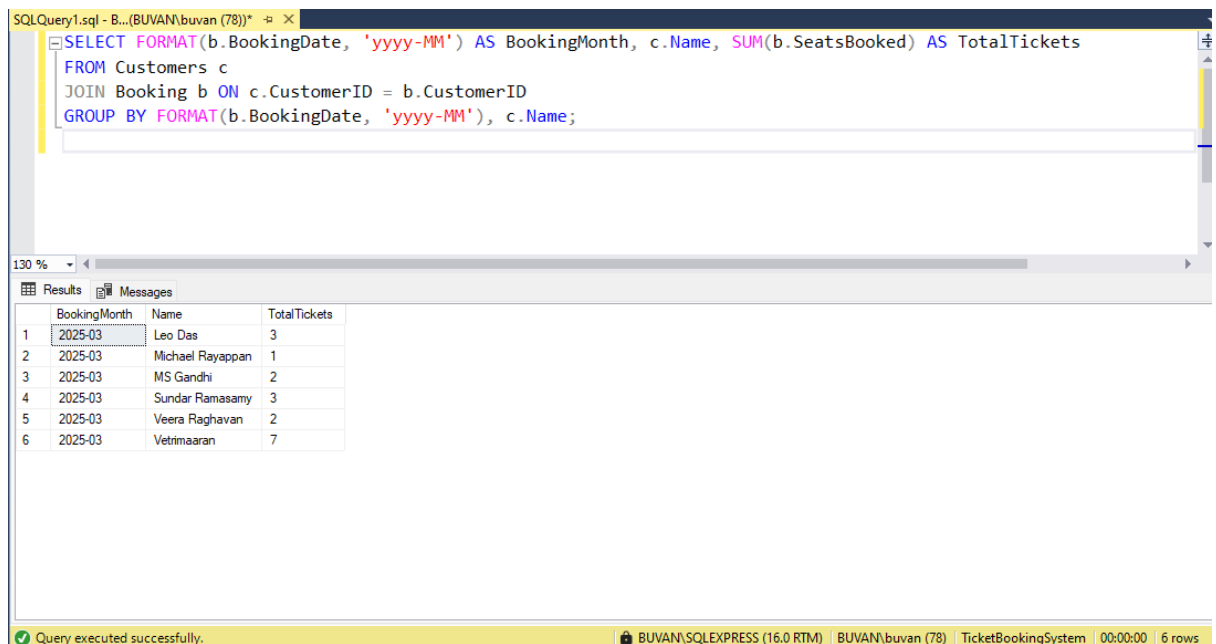
```
SELECT Category,
       (SELECT SUM(SeatsBooked)
        FROM Booking
        WHERE EventID IN (
            SELECT EventID FROM Event WHERE Category = e.Category
        )) AS TotalTicketsSold
FROM Event e
GROUP BY Category;
```

The query results are displayed in a table with 4 rows:

Category	TotalTicketsSold
1 Concert	2
2 Other	12
3 Sports	1
4 Theater	3

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 4 rows

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



The screenshot shows a SQL query window with the following query:

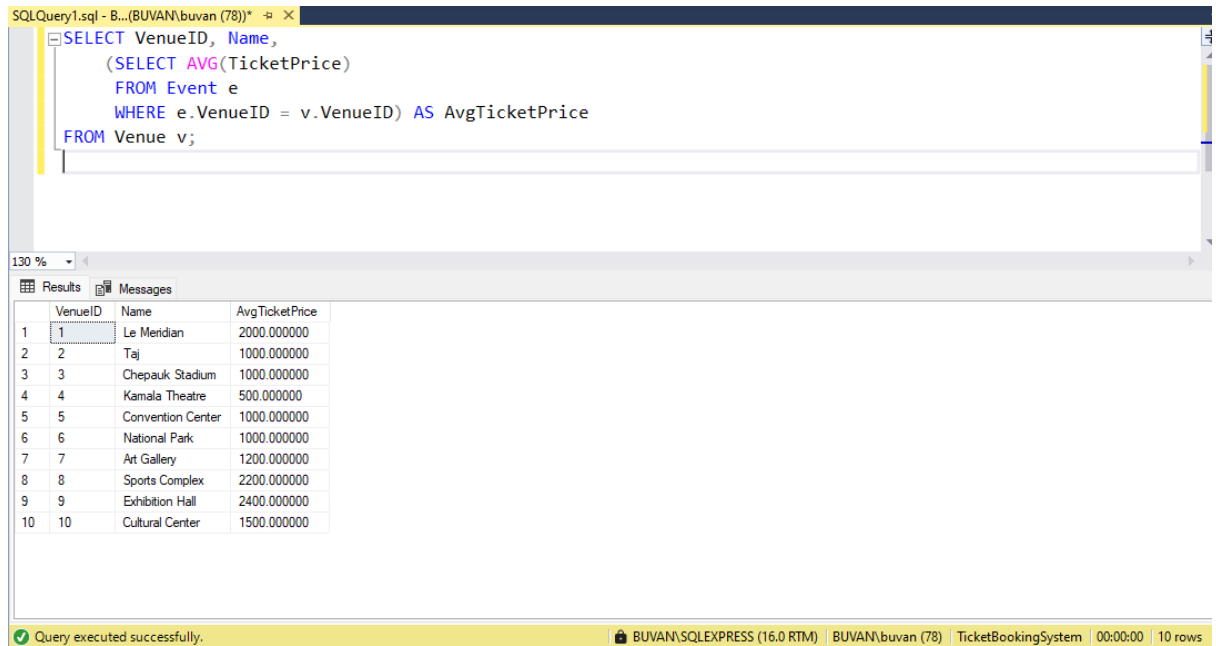
```
SELECT FORMAT(b.BookingDate, 'yyyy-MM') AS BookingMonth, c.Name, SUM(b.SeatsBooked) AS TotalTickets
FROM Customers c
JOIN Booking b ON c.CustomerID = b.CustomerID
GROUP BY FORMAT(b.BookingDate, 'yyyy-MM'), c.Name;
```

The query results are displayed in a table with 6 rows:

BookingMonth	Name	TotalTickets
2025-03	Leo Das	3
2025-03	Michael Rayappan	1
2025-03	MS Gandhi	2
2025-03	Sundar Ramasamy	3
2025-03	Veera Raghavan	2
2025-03	Vetrimaaran	7

The status bar at the bottom indicates: Query executed successfully. | BUVAN\SQLEXPRESS (16.0 RTM) | BUVAN\buvan (78) | TicketBookingSystem | 00:00:00 | 6 rows

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



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window with the following SQL code:

```
SELECT VenueID, Name,  
       (SELECT AVG(TicketPrice)  
        FROM Event e  
        WHERE e.VenueID = v.VenueID) AS AvgTicketPrice  
FROM Venue v;
```

The bottom pane shows the results of the query in a table format. The table has three columns: VenueID, Name, and AvgTicketPrice. The results are as follows:

	VenueID	Name	AvgTicketPrice
1	1	Le Meridian	2000.000000
2	2	Taj	1000.000000
3	3	Chepauk Stadium	1000.000000
4	4	Kamala Theatre	500.000000
5	5	Convention Center	1000.000000
6	6	National Park	1000.000000
7	7	Art Gallery	1200.000000
8	8	Sports Complex	2200.000000
9	9	Exhibition Hall	2400.000000
10	10	Cultural Center	1500.000000

The status bar at the bottom indicates that the query was executed successfully. The message pane shows: "Query executed successfully." The bottom right corner displays the server name "BUVAN\SQLEXPRESS (16.0 RTM)", the database name "BUVAN\buvan (78)", the system name "TicketBookingSystem", the execution time "00:00:00", and the number of rows "10 rows".