ASSIGNMENT-5(HEXA)

Task-1: By:Satyendra Singh Rathore

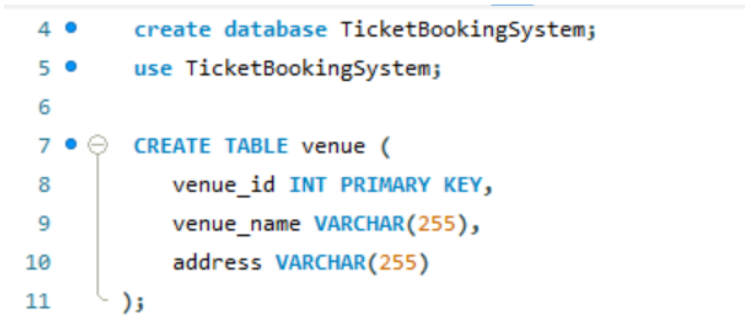
1. Create the database named "TicketBookingSystem"

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

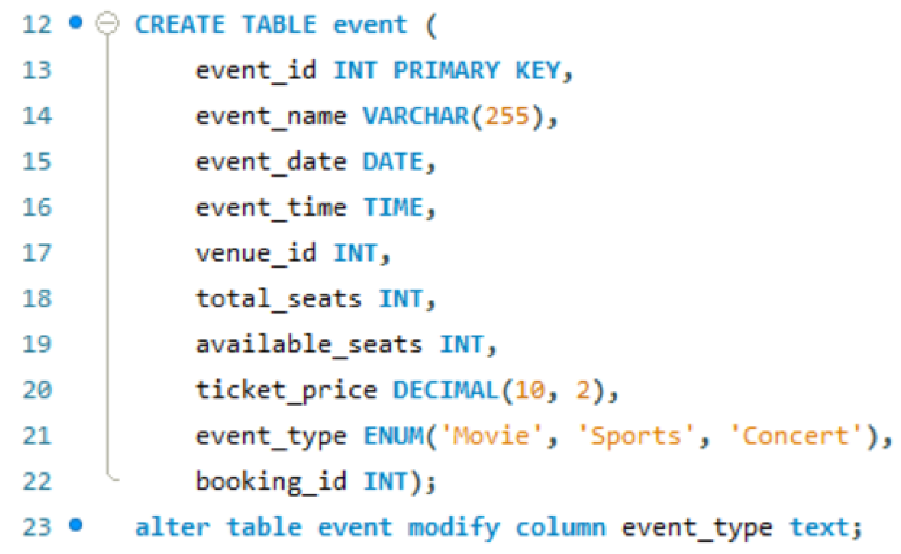
- Venue
- Event
- Customers
- Booking

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

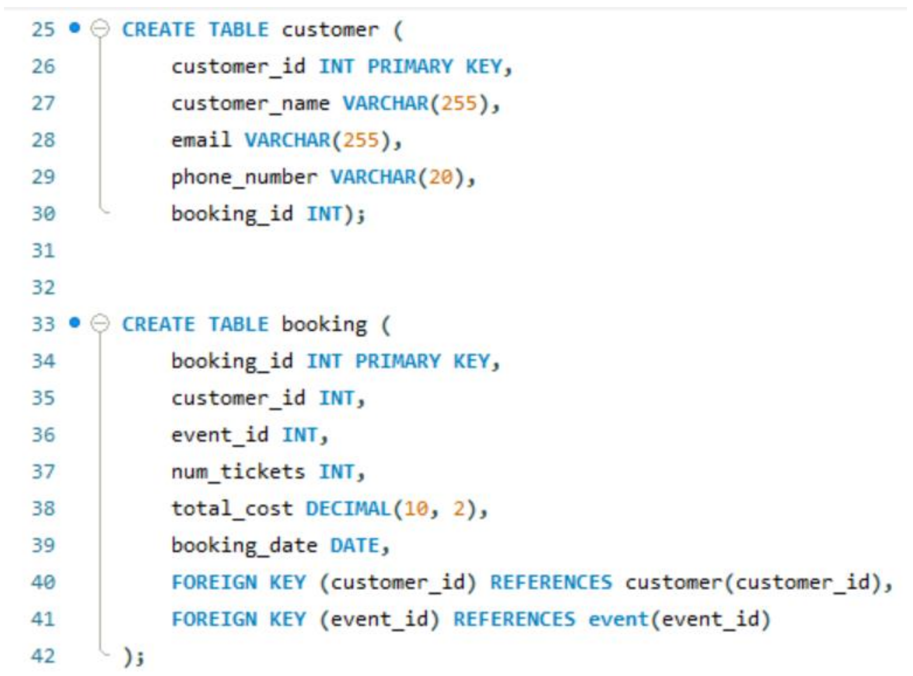
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



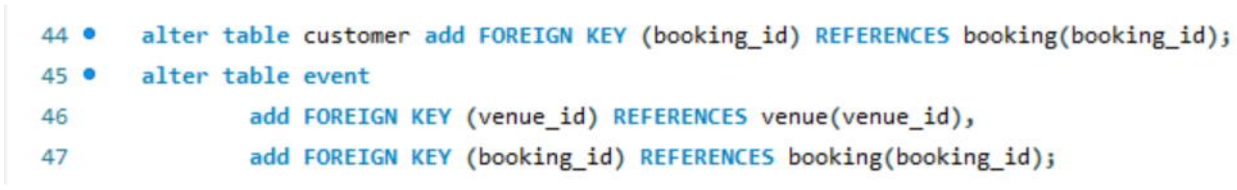
```
create database TicketBookingSystem;
use TicketBookingSystem;

CREATE TABLE venue (
    venue_id INT PRIMARY KEY,
    venue_name VARCHAR(255),
    address VARCHAR(255)
);
```



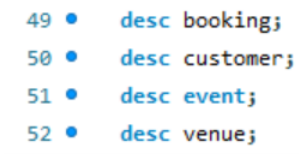
```
CREATE TABLE event (
    event_id INT PRIMARY KEY,
    event_name VARCHAR(255),
    event_date DATE,
    event_time TIME,
    venue_id INT,
    total_seats INT,
    available_seats INT,
    ticket_price DECIMAL(10, 2),
    event_type ENUM('Movie', 'Sports', 'Concert'),
    booking_id INT);
alter table event modify column event_type text;
```



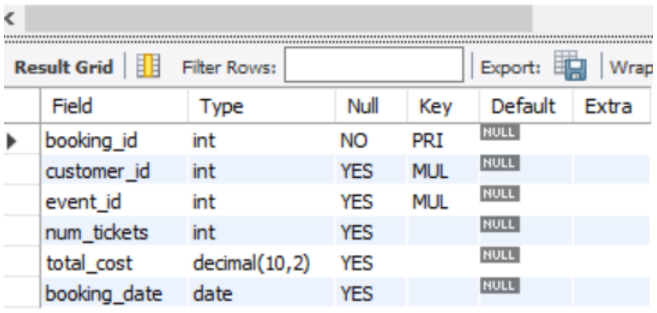
```sql
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(20),
    booking_id INT);


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,
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    FOREIGN KEY (event_id) REFERENCES event(event_id)
);
```



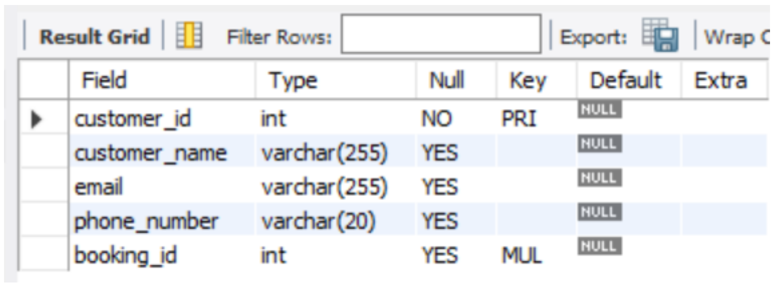
```sql
alter table customer add FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
alter table event
        add FOREIGN KEY (venue_id) REFERENCES venue(venue_id),
        add FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
```



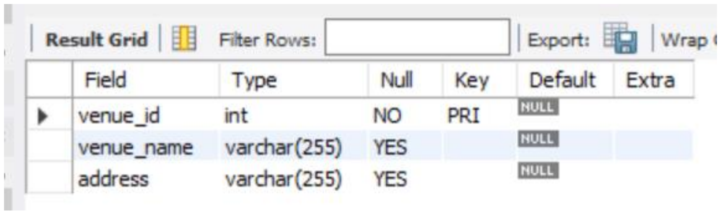
```sql
desc booking;
desc customer;
desc event;
desc venue;
```



Field	Type	Null	Key	Default	Extra
booking_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
event_id	int	YES	MUL	NULL	
num_tickets	int	YES		NULL	
total_cost	decimal(10,2)	YES		NULL	
booking_date	date	YES		NULL	

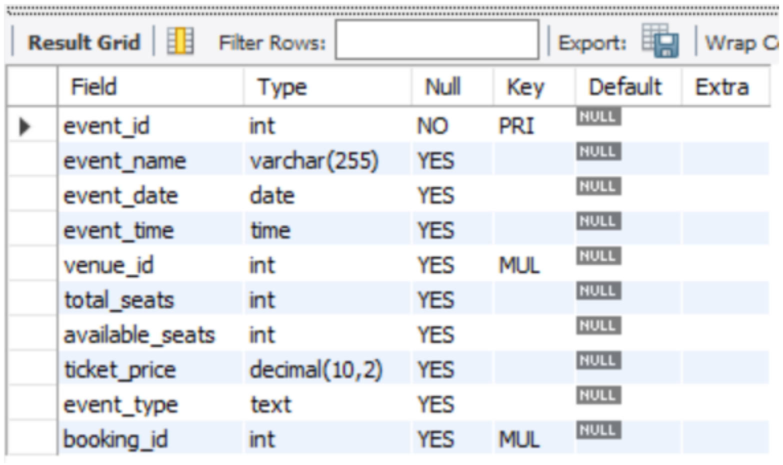


Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
customer_name	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
phone_number	varchar(20)	YES		NULL	
booking_id	int	YES	MUL	NULL	



Result Grid | Filter Rows: | Export: | Wrap

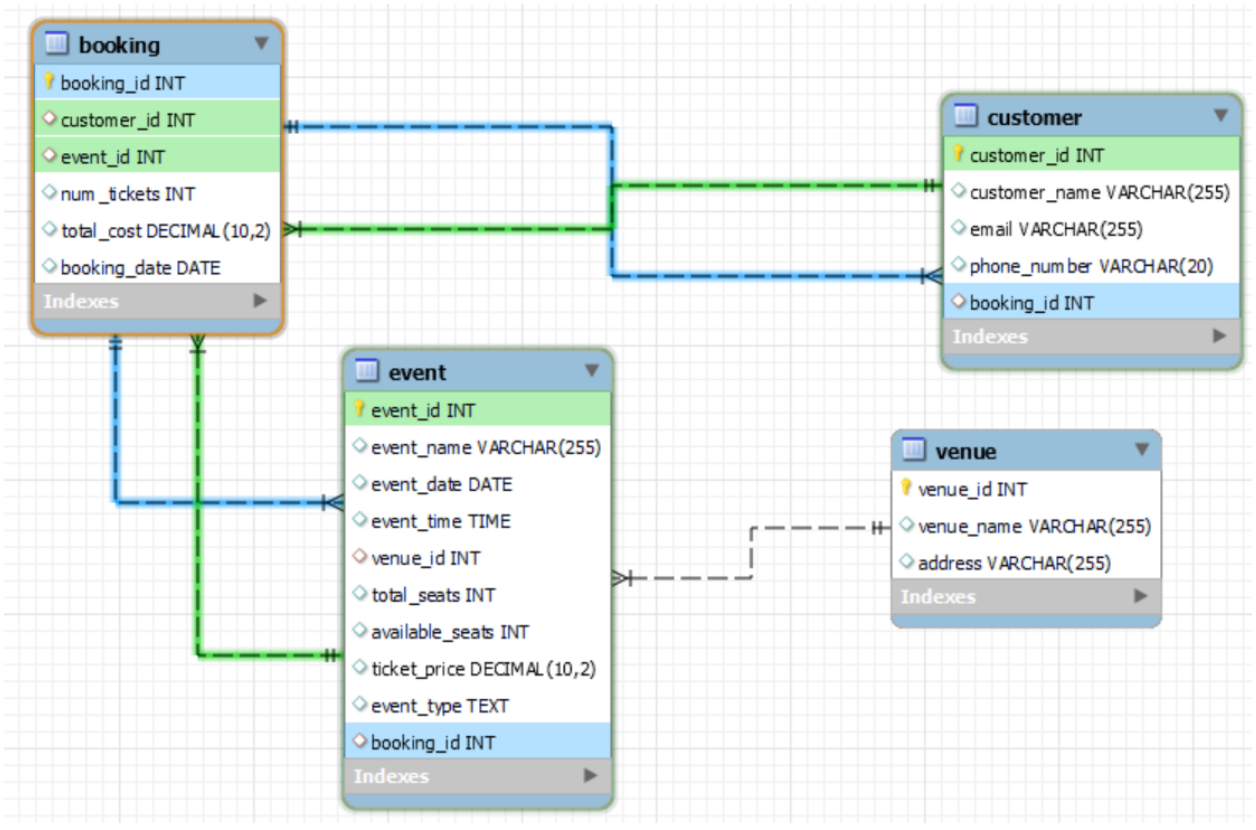
Field	Type	Null	Key	Default	Extra
venue_id	int	NO	PRI	NULL	
venue_name	varchar(255)	YES		NULL	
address	varchar(255)	YES		NULL	



Result Grid | Filter Rows: | Export: | Wrap

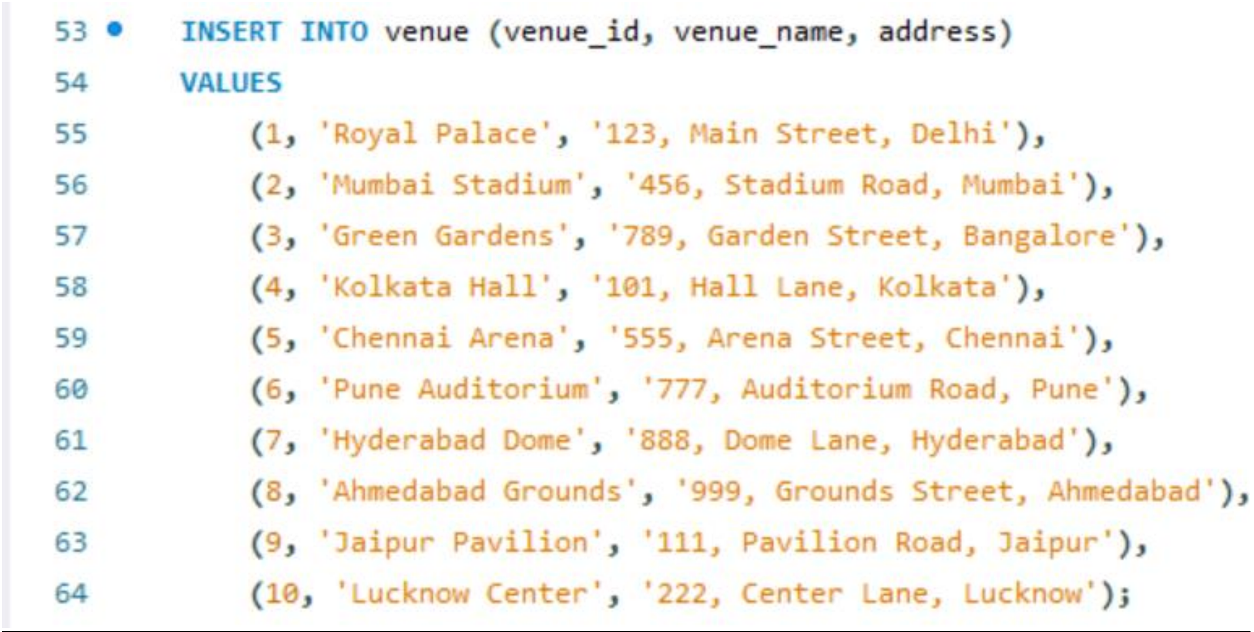
Field	Type	Null	Key	Default	Extra
event_id	int	NO	PRI	NULL	
event_name	varchar(255)	YES		NULL	
event_date	date	YES		NULL	
event_time	time	YES		NULL	
venue_id	int	YES	MUL	NULL	
total_seats	int	YES		NULL	
available_seats	int	YES		NULL	
ticket_price	decimal(10,2)	YES		NULL	
event_type	text	YES		NULL	
booking_id	int	YES	MUL	NULL	

Relationship Model/ERD:

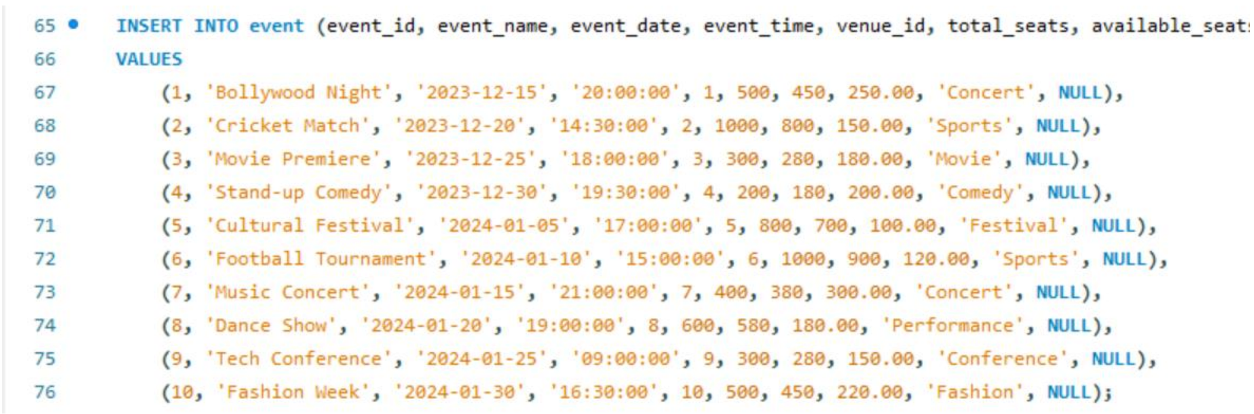


Task-2:

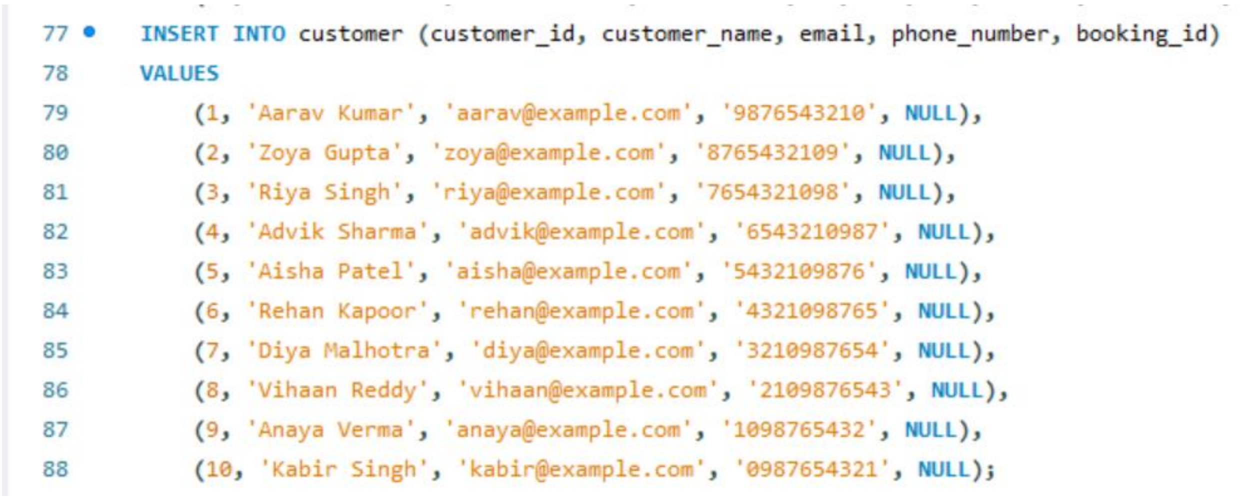
1. Write a SQL query to insert at least 10 sample records into each table.



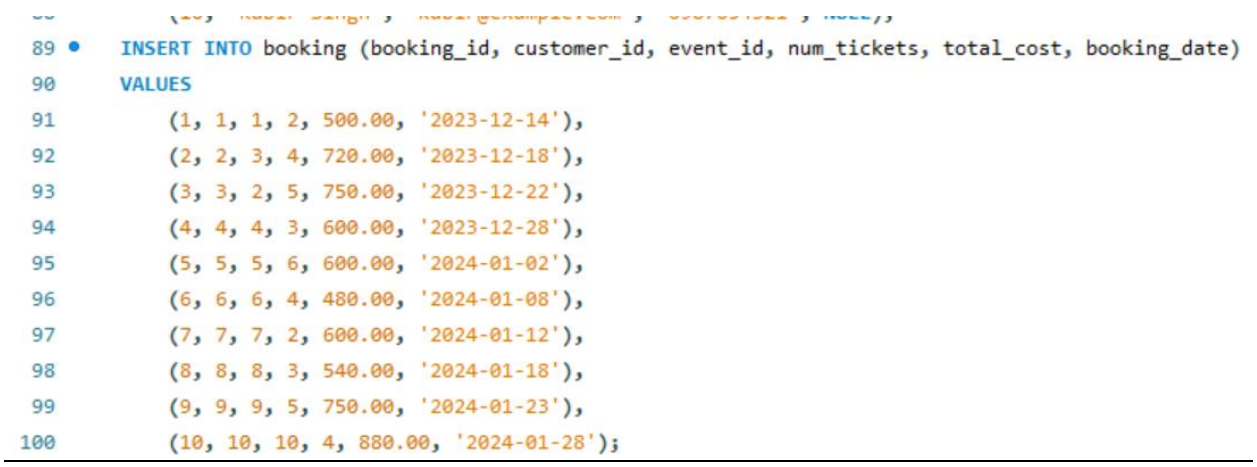
```sql
INSERT INTO venue (venue_id, venue_name, address)
VALUES
    (1, 'Royal Palace', '123, Main Street, Delhi'),
    (2, 'Mumbai Stadium', '456, Stadium Road, Mumbai'),
    (3, 'Green Gardens', '789, Garden Street, Bangalore'),
    (4, 'Kolkata Hall', '101, Hall Lane, Kolkata'),
    (5, 'Chennai Arena', '555, Arena Street, Chennai'),
    (6, 'Pune Auditorium', '777, Auditorium Road, Pune'),
    (7, 'Hyderabad Dome', '888, Dome Lane, Hyderabad'),
    (8, 'Ahmedabad Grounds', '999, Grounds Street, Ahmedabad'),
    (9, 'Jaipur Pavilion', '111, Pavilion Road, Jaipur'),
    (10, 'Lucknow Center', '222, Center Lane, Lucknow');
```



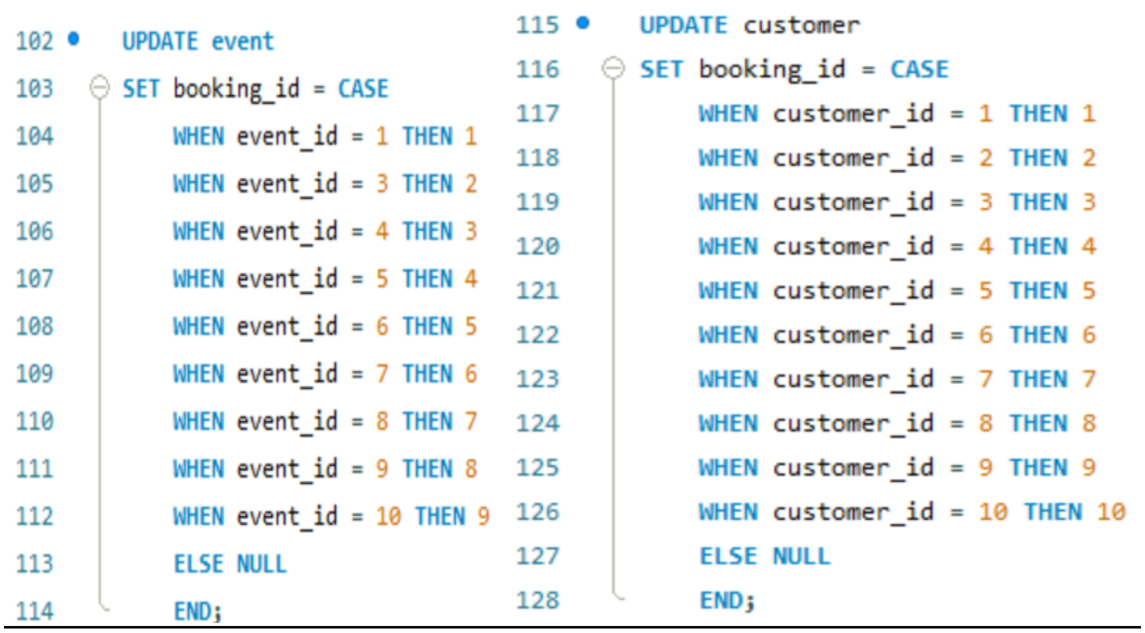
```sql
INSERT INTO event (event_id, event_name, event_date, event_time, venue_id, total_seats, available_seat
VALUES
    (1, 'Bollywood Night', '2023-12-15', '20:00:00', 1, 500, 450, 250.00, 'Concert', NULL),
    (2, 'Cricket Match', '2023-12-20', '14:30:00', 2, 1000, 800, 150.00, 'Sports', NULL),
    (3, 'Movie Premiere', '2023-12-25', '18:00:00', 3, 300, 280, 180.00, 'Movie', NULL),
    (4, 'Stand-up Comedy', '2023-12-30', '19:30:00', 4, 200, 180, 200.00, 'Comedy', NULL),
    (5, 'Cultural Festival', '2024-01-05', '17:00:00', 5, 800, 700, 100.00, 'Festival', NULL),
    (6, 'Football Tournament', '2024-01-10', '15:00:00', 6, 1000, 900, 120.00, 'Sports', NULL),
    (7, 'Music Concert', '2024-01-15', '21:00:00', 7, 400, 380, 300.00, 'Concert', NULL),
    (8, 'Dance Show', '2024-01-20', '19:00:00', 8, 600, 580, 180.00, 'Performance', NULL),
    (9, 'Tech Conference', '2024-01-25', '09:00:00', 9, 300, 280, 150.00, 'Conference', NULL),
    (10, 'Fashion Week', '2024-01-30', '16:30:00', 10, 500, 450, 220.00, 'Fashion', NULL);
```



```sql
INSERT INTO customer (customer_id, customer_name, email, phone_number, booking_id)
VALUES
    (1, 'Aarav Kumar', 'aarav@example.com', '9876543210', NULL),
    (2, 'Zoya Gupta', 'zoya@example.com', '8765432109', NULL),
    (3, 'Riya Singh', 'riya@example.com', '7654321098', NULL),
    (4, 'Advik Sharma', 'advik@example.com', '6543210987', NULL),
    (5, 'Aisha Patel', 'aisha@example.com', '5432109876', NULL),
    (6, 'Rehan Kapoor', 'rehan@example.com', '4321098765', NULL),
    (7, 'Diya Malhotra', 'diya@example.com', '3210987654', NULL),
    (8, 'Vihaan Reddy', 'vihaan@example.com', '2109876543', NULL),
    (9, 'Anaya Verma', 'anaya@example.com', '1098765432', NULL),
    (10, 'Kabir Singh', 'kabir@example.com', '0987654321', NULL);
```

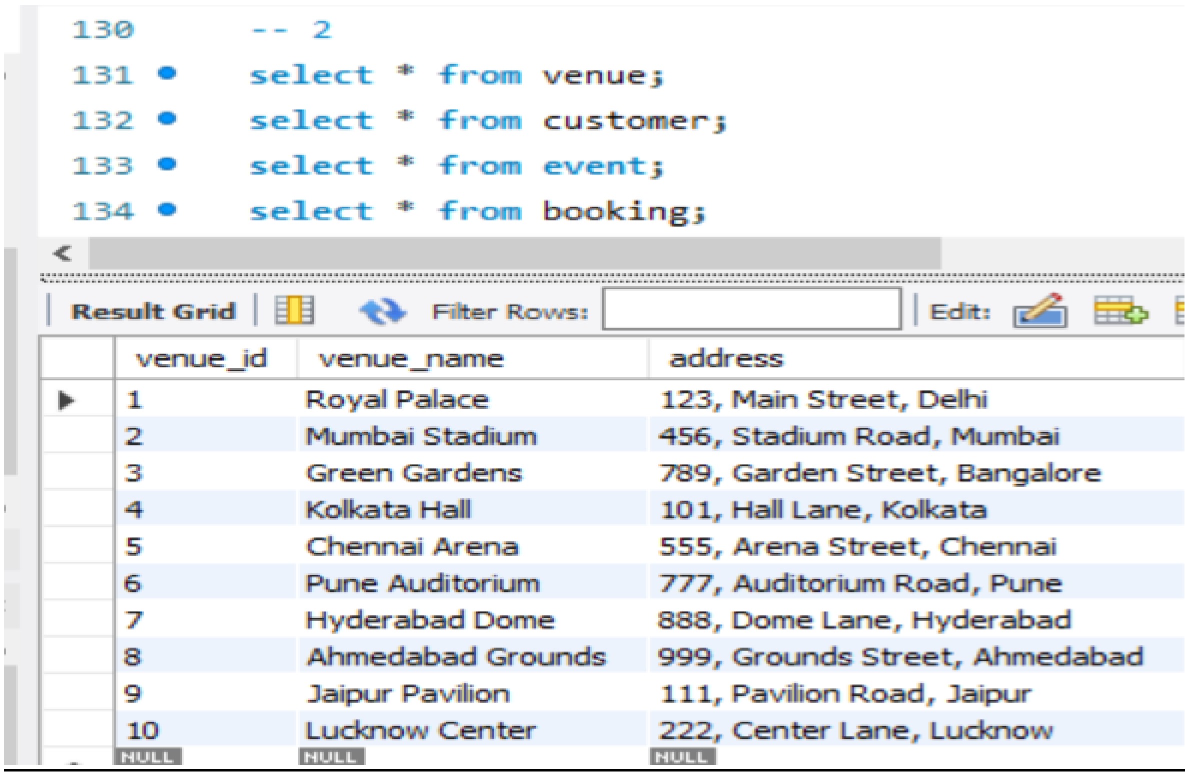


```sql
INSERT INTO booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date)
VALUES
    (1, 1, 1, 2, 500.00, '2023-12-14'),
    (2, 2, 3, 4, 720.00, '2023-12-18'),
    (3, 3, 2, 5, 750.00, '2023-12-22'),
    (4, 4, 4, 3, 600.00, '2023-12-28'),
    (5, 5, 5, 6, 600.00, '2024-01-02'),
    (6, 6, 6, 4, 480.00, '2024-01-08'),
    (7, 7, 7, 2, 600.00, '2024-01-12'),
    (8, 8, 8, 3, 540.00, '2024-01-18'),
    (9, 9, 9, 5, 750.00, '2024-01-23'),
    (10, 10, 10, 4, 880.00, '2024-01-28');
```



```sql
UPDATE event
SET booking_id = CASE
    WHEN event_id = 1 THEN 1
    WHEN event_id = 3 THEN 2
    WHEN event_id = 4 THEN 3
    WHEN event_id = 5 THEN 4
    WHEN event_id = 6 THEN 5
    WHEN event_id = 7 THEN 6
    WHEN event_id = 8 THEN 7
    WHEN event_id = 9 THEN 8
    WHEN event_id = 10 THEN 9
    ELSE NULL
    END;
```

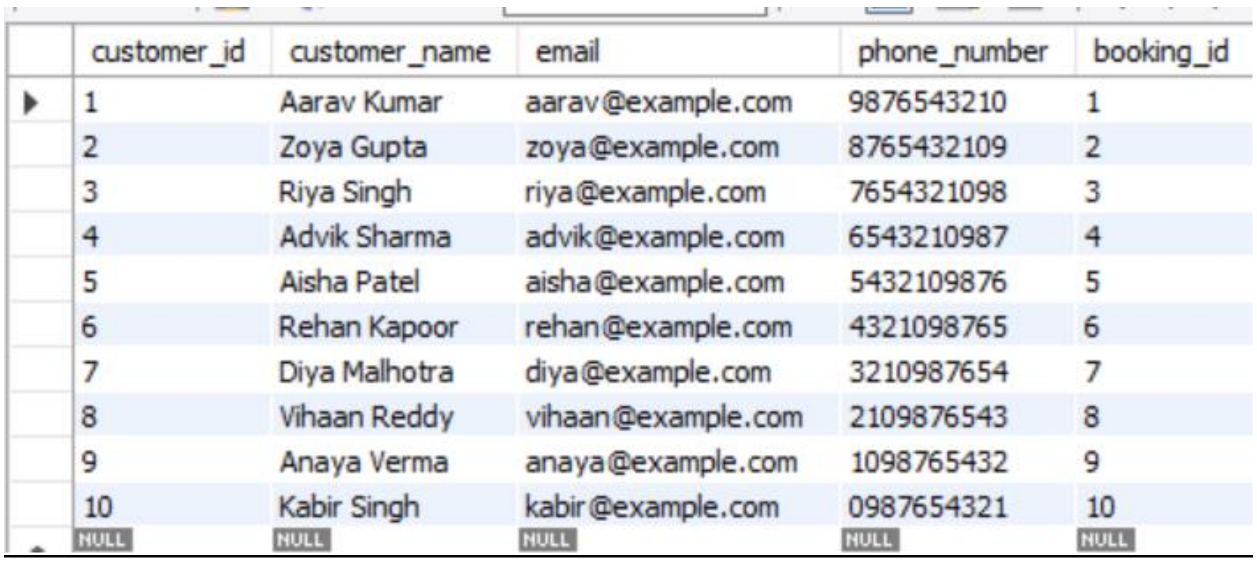
```sql
UPDATE customer
SET booking_id = CASE
    WHEN customer_id = 1 THEN 1
    WHEN customer_id = 2 THEN 2
    WHEN customer_id = 3 THEN 3
    WHEN customer_id = 4 THEN 4
    WHEN customer_id = 5 THEN 5
    WHEN customer_id = 6 THEN 6
    WHEN customer_id = 7 THEN 7
    WHEN customer_id = 8 THEN 8
    WHEN customer_id = 9 THEN 9
    WHEN customer_id = 10 THEN 10
    ELSE NULL
    END;
```



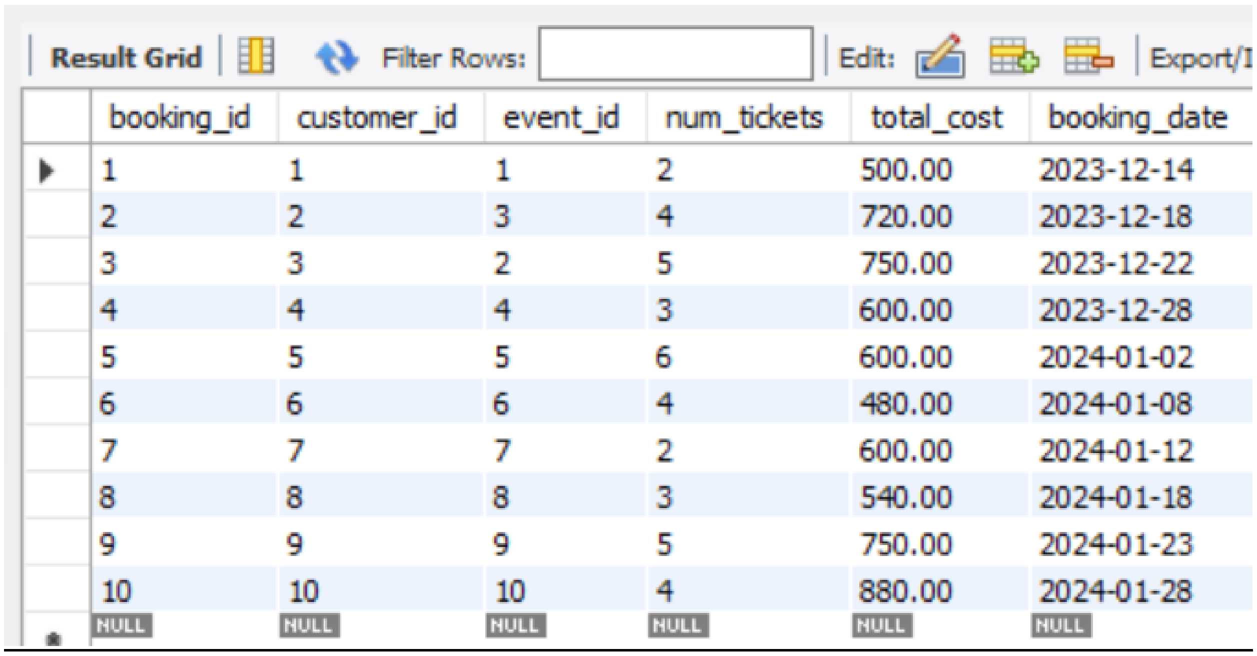
```
-- 2
select * from venue;
select * from customer;
select * from event;
select * from booking;
```

Result Grid | Filter Rows: | Edit:

venue_id	venue_name	address
1	Royal Palace	123, Main Street, Delhi
2	Mumbai Stadium	456, Stadium Road, Mumbai
3	Green Gardens	789, Garden Street, Bangalore
4	Kolkata Hall	101, Hall Lane, Kolkata
5	Chennai Arena	555, Arena Street, Chennai
6	Pune Auditorium	777, Auditorium Road, Pune
7	Hyderabad Dome	888, Dome Lane, Hyderabad
8	Ahmedabad Grounds	999, Grounds Street, Ahmedabad
9	Jaipur Pavilion	111, Pavilion Road, Jaipur
10	Lucknow Center	222, Center Lane, Lucknow
NULL	NULL	NULL

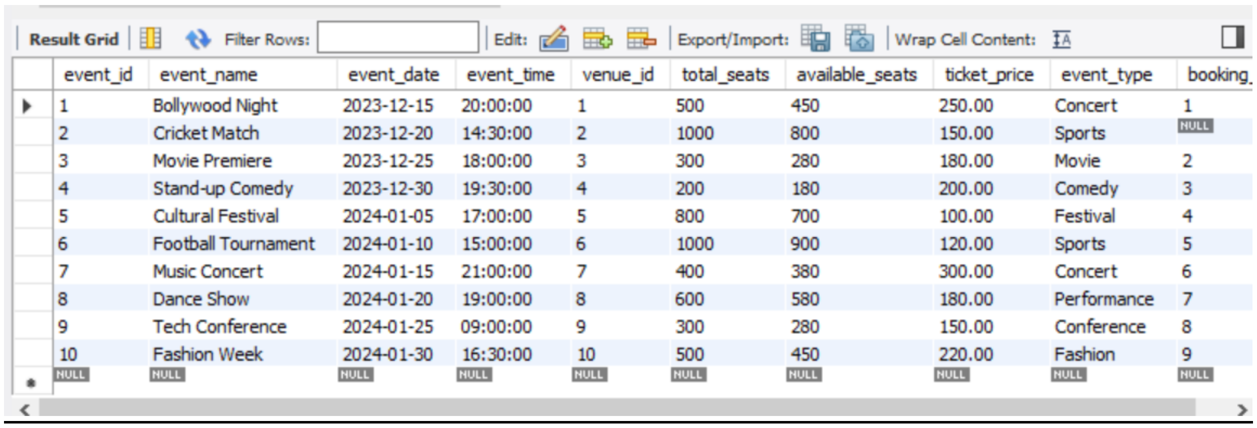


customer_id	customer_name	email	phone_number	booking_id
1	Aarav Kumar	aarav@example.com	9876543210	1
2	Zoya Gupta	zoya@example.com	8765432109	2
3	Riya Singh	riya@example.com	7654321098	3
4	Advik Sharma	advik@example.com	6543210987	4
5	Aisha Patel	aisha@example.com	5432109876	5
6	Rehan Kapoor	rehan@example.com	4321098765	6
7	Diya Malhotra	diya@example.com	3210987654	7
8	Vihaan Reddy	vihaan@example.com	2109876543	8
9	Anaya Verma	anaya@example.com	1098765432	9
10	Kabir Singh	kabir@example.com	0987654321	10
NULL	NULL	NULL	NULL	NULL



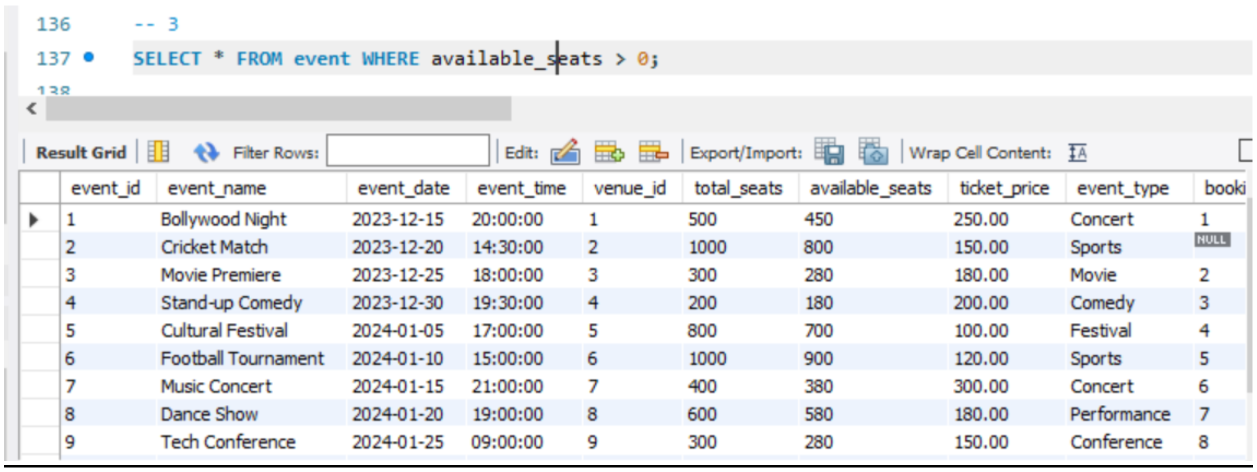
booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1	1	1	2	500.00	2023-12-14
2	2	3	4	720.00	2023-12-18
3	3	2	5	750.00	2023-12-22
4	4	4	3	600.00	2023-12-28
5	5	5	6	600.00	2024-01-02
6	6	6	4	480.00	2024-01-08
7	7	7	2	600.00	2024-01-12
8	8	8	3	540.00	2024-01-18
9	9	9	5	750.00	2024-01-23
10	10	10	4	880.00	2024-01-28
NULL	NULL	NULL	NULL	NULL	NULL

2. Write a SQL query to list all Events.



event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_
1	Bollywood Night	2023-12-15	20:00:00	1	500	450	250.00	Concert	1
2	Cricket Match	2023-12-20	14:30:00	2	1000	800	150.00	Sports	NULL
3	Movie Premiere	2023-12-25	18:00:00	3	300	280	180.00	Movie	2
4	Stand-up Comedy	2023-12-30	19:30:00	4	200	180	200.00	Comedy	3
5	Cultural Festival	2024-01-05	17:00:00	5	800	700	100.00	Festival	4
6	Football Tournament	2024-01-10	15:00:00	6	1000	900	120.00	Sports	5
7	Music Concert	2024-01-15	21:00:00	7	400	380	300.00	Concert	6
8	Dance Show	2024-01-20	19:00:00	8	600	580	180.00	Performance	7
9	Tech Conference	2024-01-25	09:00:00	9	300	280	150.00	Conference	8
10	Fashion Week	2024-01-30	16:30:00	10	500	450	220.00	Fashion	9
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

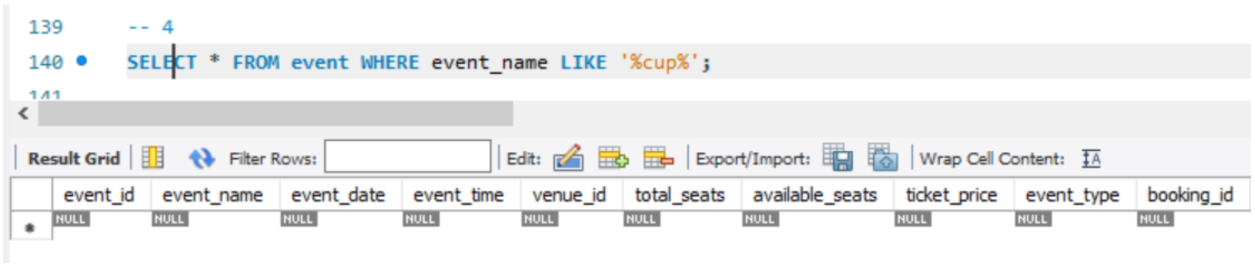
3. Write a SQL query to select events with available tickets.



```
-- 3
SELECT * FROM event WHERE available_seats > 0;
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booki
1	Bollywood Night	2023-12-15	20:00:00	1	500	450	250.00	Concert	1
2	Cricket Match	2023-12-20	14:30:00	2	1000	800	150.00	Sports	NULL
3	Movie Premiere	2023-12-25	18:00:00	3	300	280	180.00	Movie	2
4	Stand-up Comedy	2023-12-30	19:30:00	4	200	180	200.00	Comedy	3
5	Cultural Festival	2024-01-05	17:00:00	5	800	700	100.00	Festival	4
6	Football Tournament	2024-01-10	15:00:00	6	1000	900	120.00	Sports	5
7	Music Concert	2024-01-15	21:00:00	7	400	380	300.00	Concert	6
8	Dance Show	2024-01-20	19:00:00	8	600	580	180.00	Performance	7
9	Tech Conference	2024-01-25	09:00:00	9	300	280	150.00	Conference	8

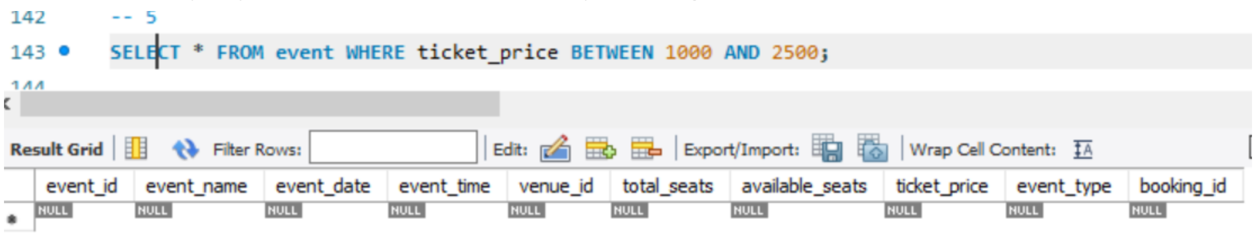
4. Write a SQL query to select events name partial match with ‘cup’.



```sql
-- 4
SELECT * FROM event WHERE event_name LIKE '%cup%';
```

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

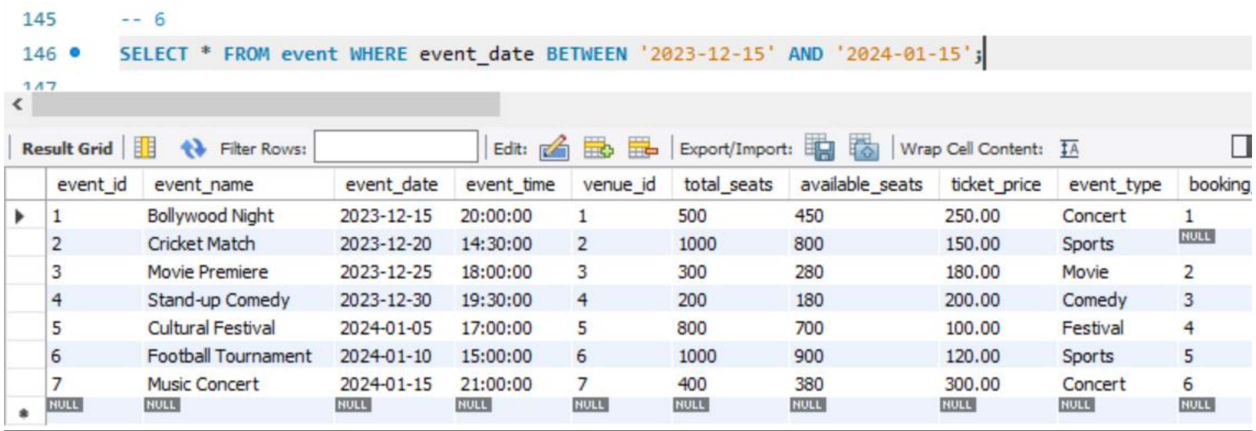
5. Write a SQL query to select events with ticket price range is between 1000 to 2500.



```sql
-- 5
SELECT * FROM event WHERE ticket_price BETWEEN 1000 AND 2500;
```

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

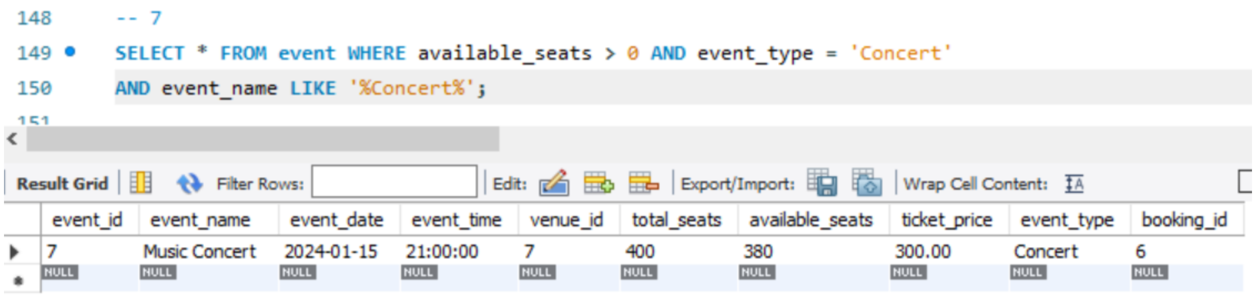
6. Write a SQL query to retrieve events with dates falling within a specific range.



```sql
-- 6
SELECT * FROM event WHERE event_date BETWEEN '2023-12-15' AND '2024-01-15';
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking
1	Bollywood Night	2023-12-15	20:00:00	1	500	450	250.00	Concert	1
2	Cricket Match	2023-12-20	14:30:00	2	1000	800	150.00	Sports	NULL
3	Movie Premiere	2023-12-25	18:00:00	3	300	280	180.00	Movie	2
4	Stand-up Comedy	2023-12-30	19:30:00	4	200	180	200.00	Comedy	3
5	Cultural Festival	2024-01-05	17:00:00	5	800	700	100.00	Festival	4
6	Football Tournament	2024-01-10	15:00:00	6	1000	900	120.00	Sports	5
7	Music Concert	2024-01-15	21:00:00	7	400	380	300.00	Concert	6
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

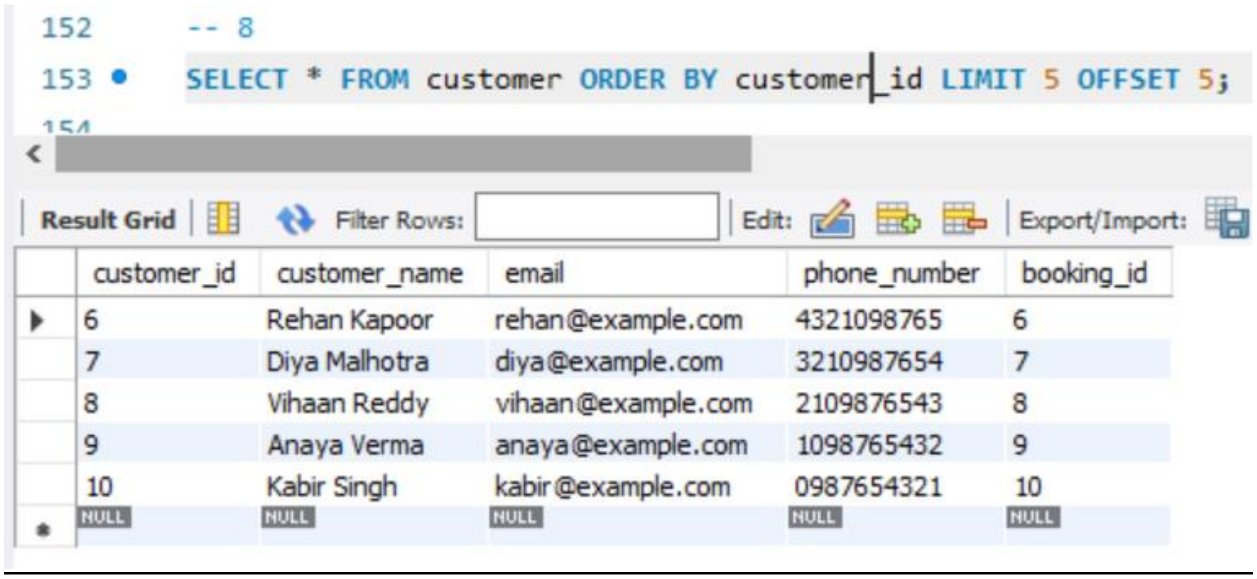
7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.



```sql
-- 7
SELECT * FROM event WHERE available_seats > 0 AND event_type = 'Concert'
AND event_name LIKE '%Concert%';
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
7	Music Concert	2024-01-15	21:00:00	7	400	380	300.00	Concert	6
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

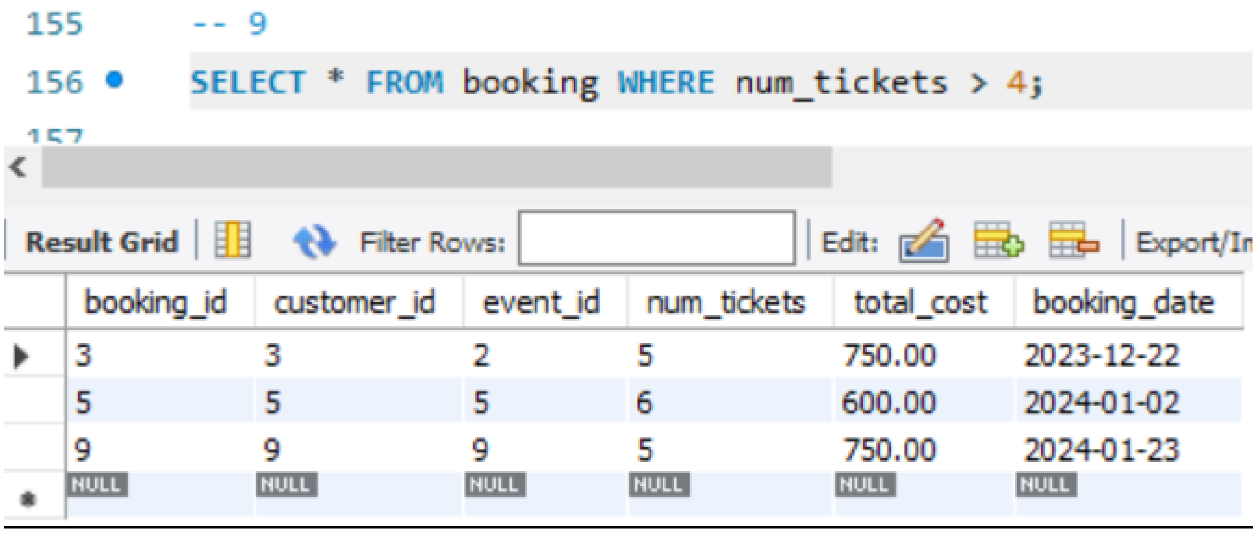
8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.



```
-- 8
SELECT * FROM customer ORDER BY customer_id LIMIT 5 OFFSET 5;
```

customer_id	customer_name	email	phone_number	booking_id
6	Rehan Kapoor	rehan@example.com	4321098765	6
7	Diya Malhotra	diya@example.com	3210987654	7
8	Vihaan Reddy	vihaan@example.com	2109876543	8
9	Anaya Verma	anaya@example.com	1098765432	9
10	Kabir Singh	kabir@example.com	0987654321	10
NULL	NULL	NULL	NULL	NULL

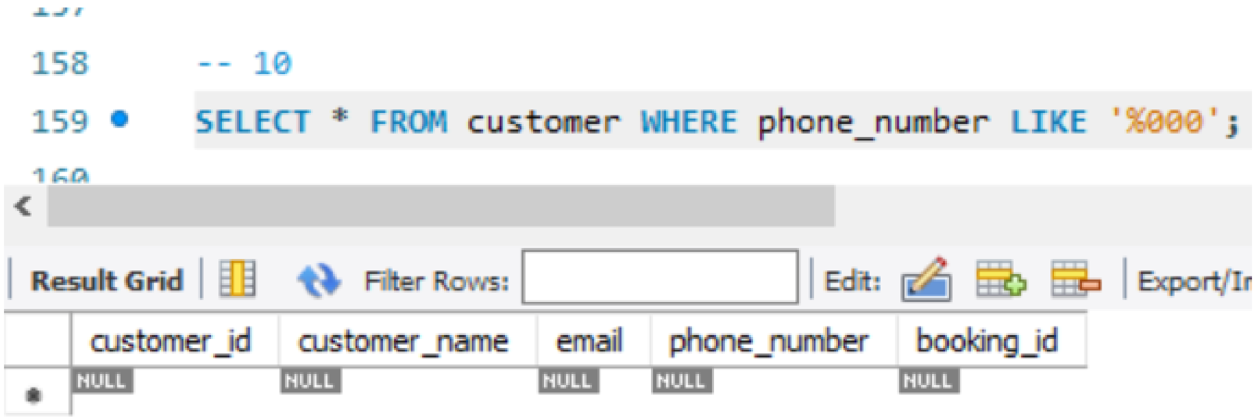
9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.



```
-- 9
SELECT * FROM booking WHERE num_tickets > 4;
```

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
3	3	2	5	750.00	2023-12-22
5	5	5	6	600.00	2024-01-02
9	9	9	5	750.00	2024-01-23
NULL	NULL	NULL	NULL	NULL	NULL

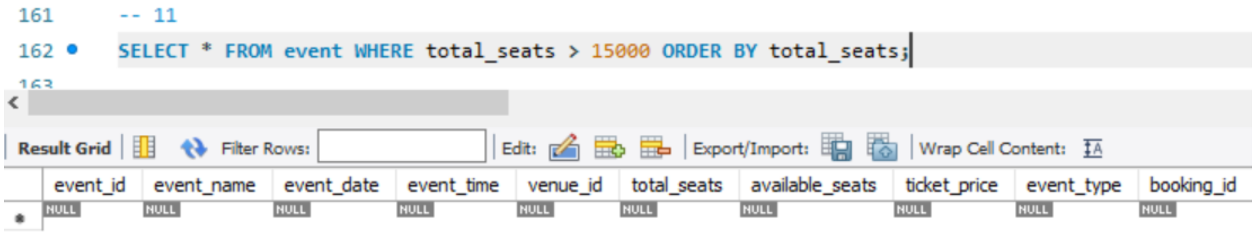
10. Write a SQL query to retrieve customer information whose phone number end with ‘000’



```sql
-- 10
SELECT * FROM customer WHERE phone_number LIKE '%000';
```

customer_id	customer_name	email	phone_number	booking_id
NULL	NULL	NULL	NULL	NULL

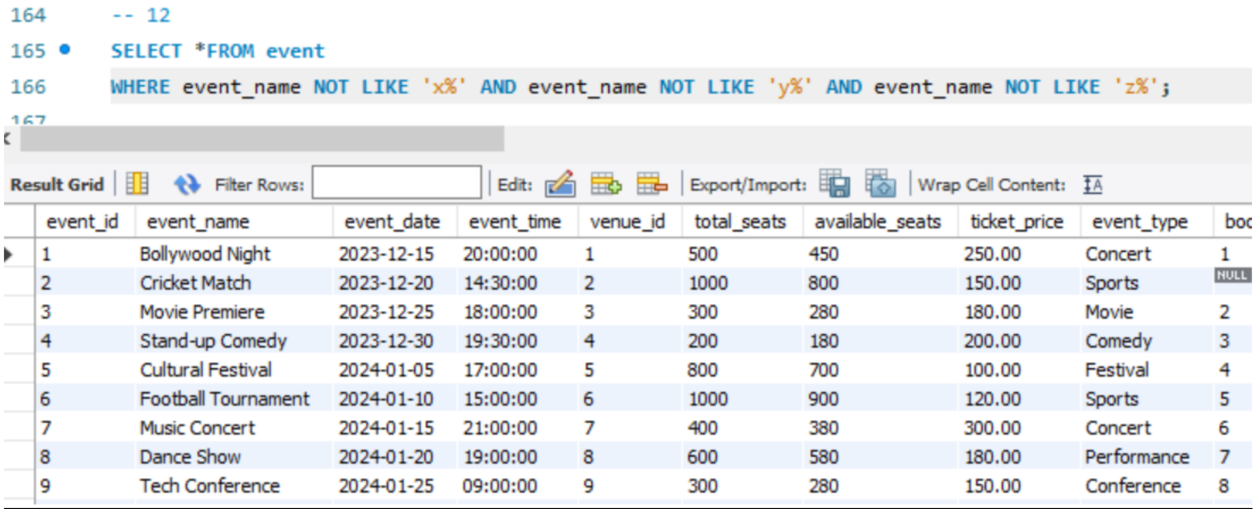
11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.



```sql
-- 11
SELECT * FROM event WHERE total_seats > 15000 ORDER BY total_seats;
```

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

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

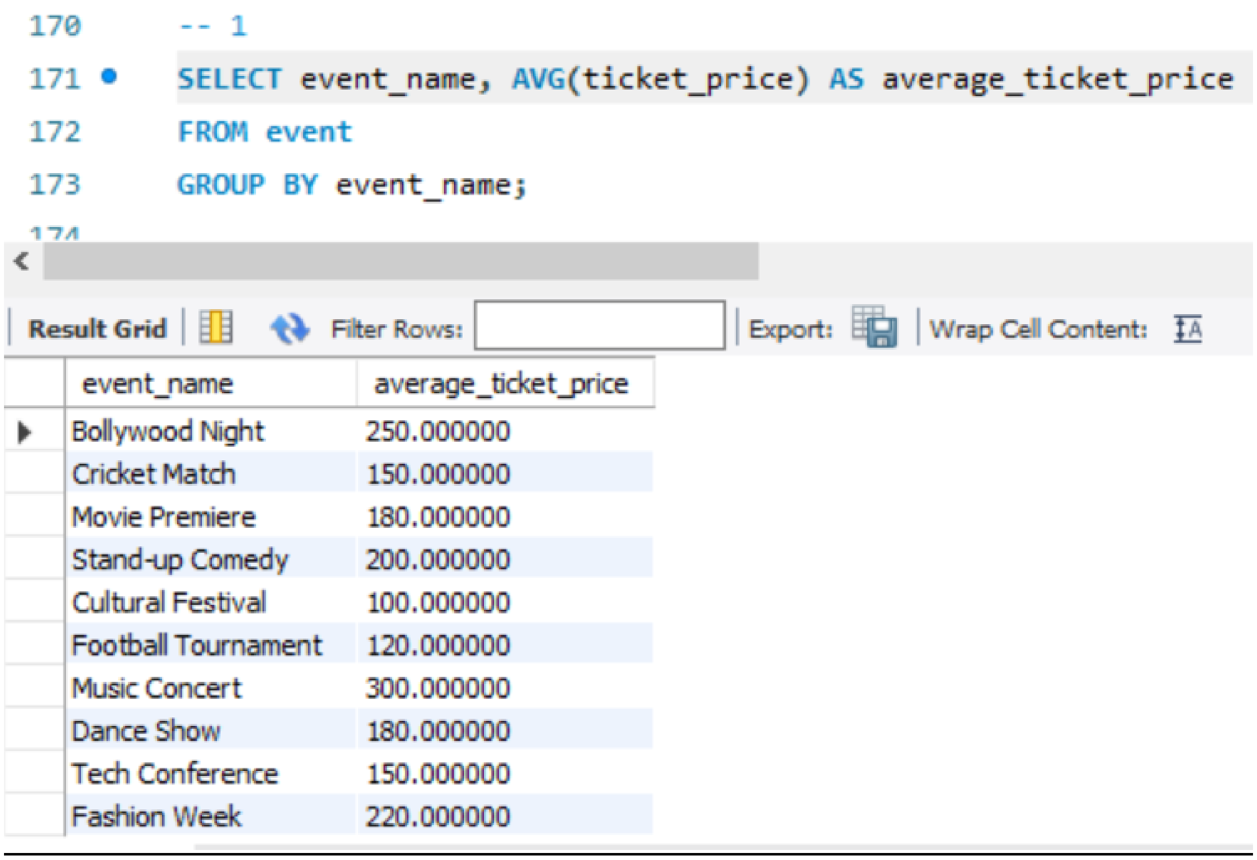


```sql
-- 12
SELECT *FROM event
WHERE event_name NOT LIKE 'x%' AND event_name NOT LIKE 'y%' AND event_name NOT LIKE 'z%';
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	bo
1	Bollywood Night	2023-12-15	20:00:00	1	500	450	250.00	Concert	1
2	Cricket Match	2023-12-20	14:30:00	2	1000	800	150.00	Sports	NULL
3	Movie Premiere	2023-12-25	18:00:00	3	300	280	180.00	Movie	2
4	Stand-up Comedy	2023-12-30	19:30:00	4	200	180	200.00	Comedy	3
5	Cultural Festival	2024-01-05	17:00:00	5	800	700	100.00	Festival	4
6	Football Tournament	2024-01-10	15:00:00	6	1000	900	120.00	Sports	5
7	Music Concert	2024-01-15	21:00:00	7	400	380	300.00	Concert	6
8	Dance Show	2024-01-20	19:00:00	8	600	580	180.00	Performance	7
9	Tech Conference	2024-01-25	09:00:00	9	300	280	150.00	Conference	8

Task-3:

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

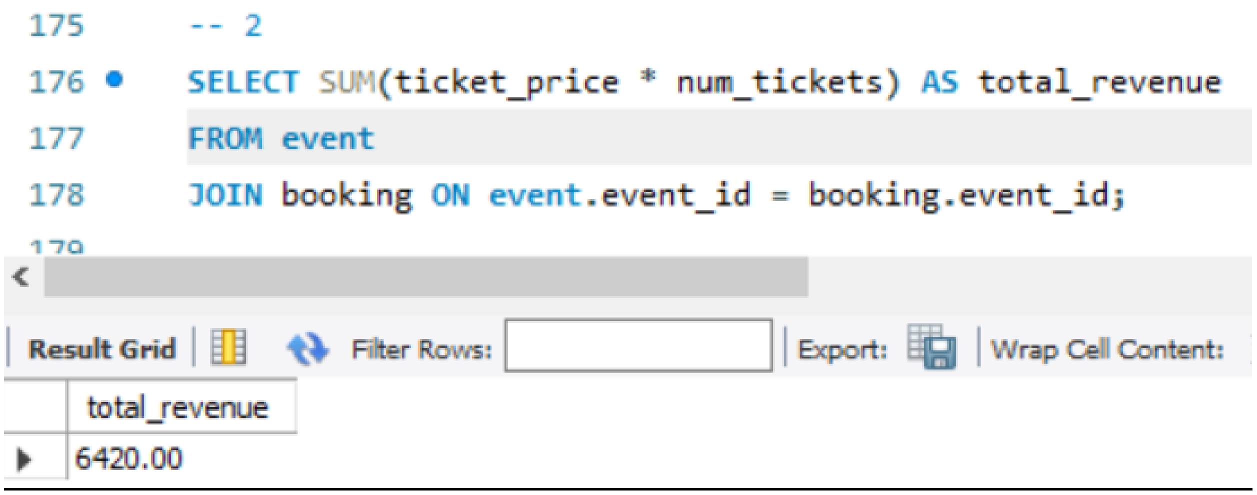


```sql
-- 1
SELECT event_name, AVG(ticket_price) AS average_ticket_price
FROM event
GROUP BY event_name;
```

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

event_name	average_ticket_price
Bollywood Night	250.000000
Cricket Match	150.000000
Movie Premiere	180.000000
Stand-up Comedy	200.000000
Cultural Festival	100.000000
Football Tournament	120.000000
Music Concert	300.000000
Dance Show	180.000000
Tech Conference	150.000000
Fashion Week	220.000000

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

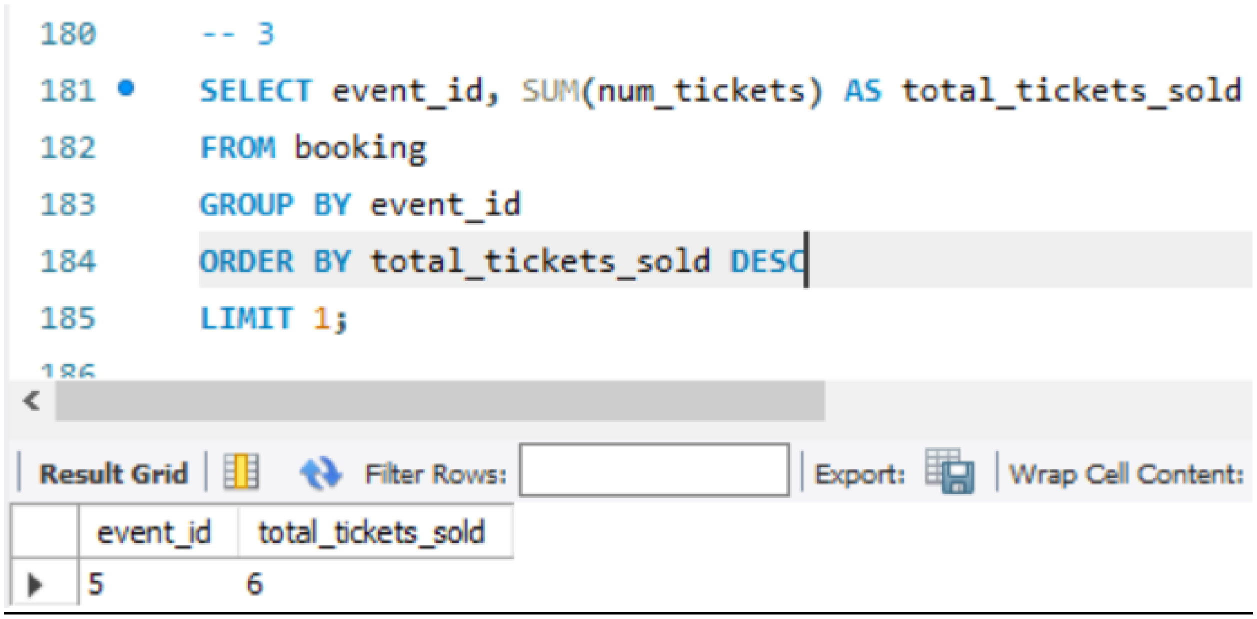


```sql
-- 2
SELECT SUM(ticket_price * num_tickets) AS total_revenue
FROM event
JOIN booking ON event.event_id = booking.event_id;
```

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

total_revenue
6420.00

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

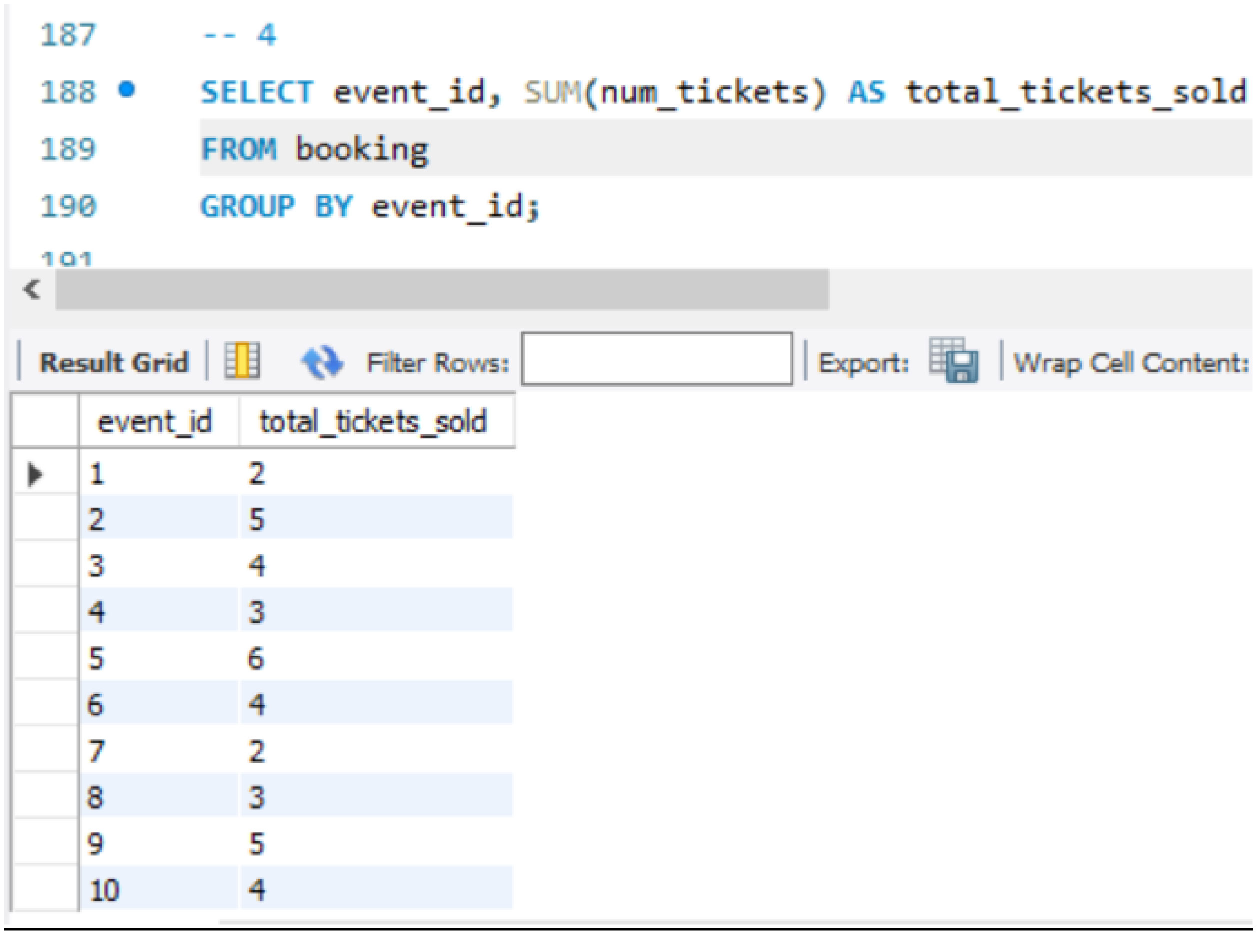


```
-- 3
SELECT event_id, SUM(num_tickets) AS total_tickets_sold
FROM booking
GROUP BY event_id
ORDER BY total_tickets_sold DESC
LIMIT 1;
```

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

event_id	total_tickets_sold
5	6

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

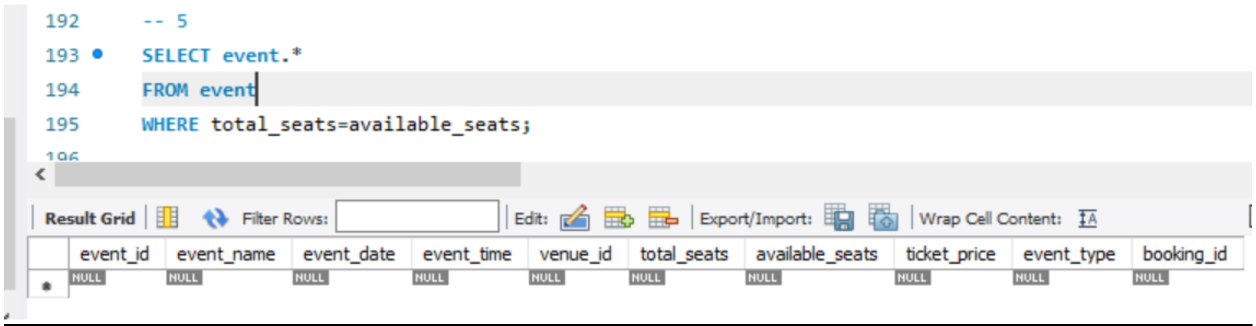


```
-- 4
SELECT event_id, SUM(num_tickets) AS total_tickets_sold
FROM booking
GROUP BY event_id;
```

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

event_id	total_tickets_sold
1	2
2	5
3	4
4	3
5	6
6	4
7	2
8	3
9	5
10	4

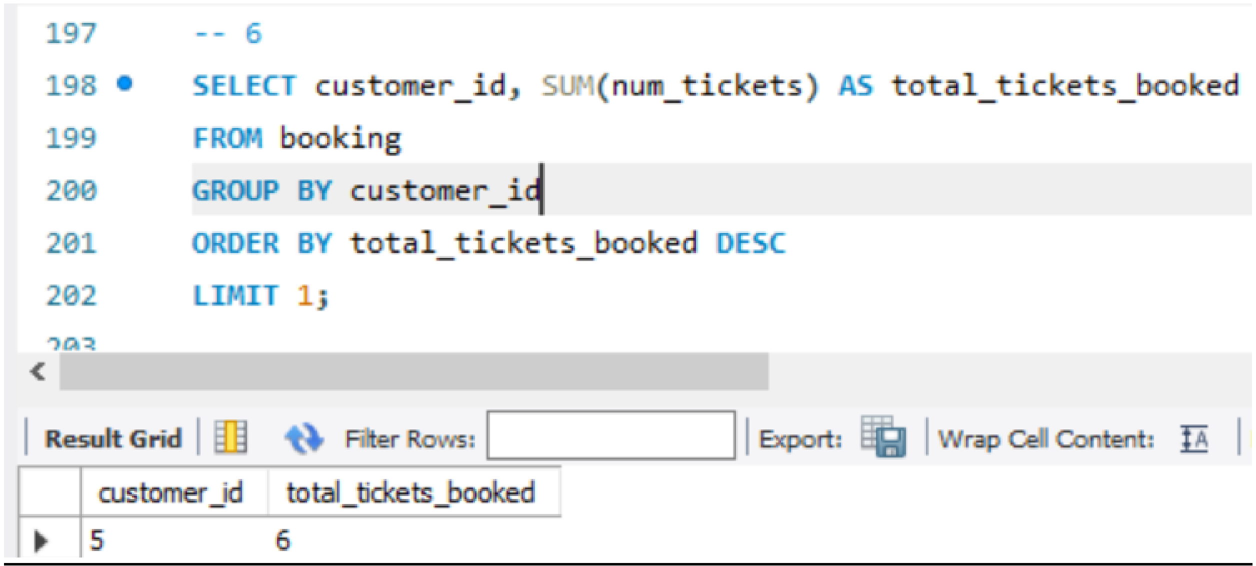
5. Write a SQL query to Find Events with No Ticket Sales.



```sql
-- 5
SELECT event.*
FROM event
WHERE total_seats=available_seats;
```

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

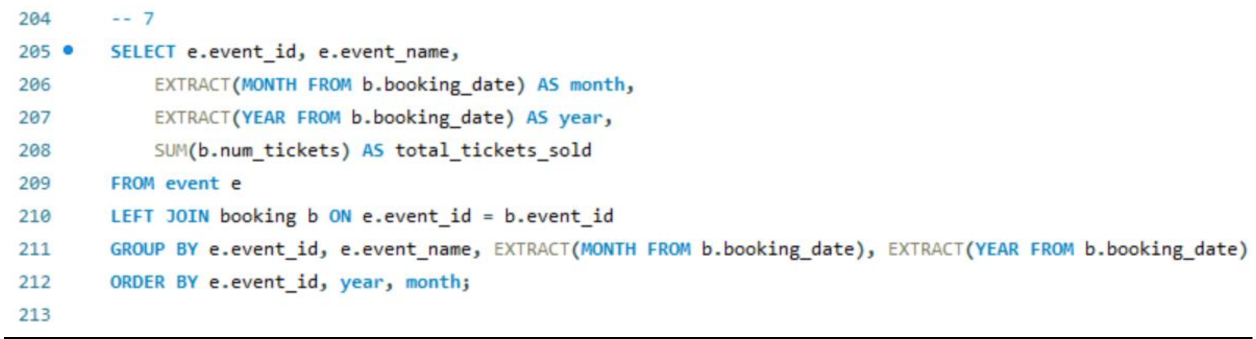
6. Write a SQL query to Find the User Who Has Booked the Most Tickets.



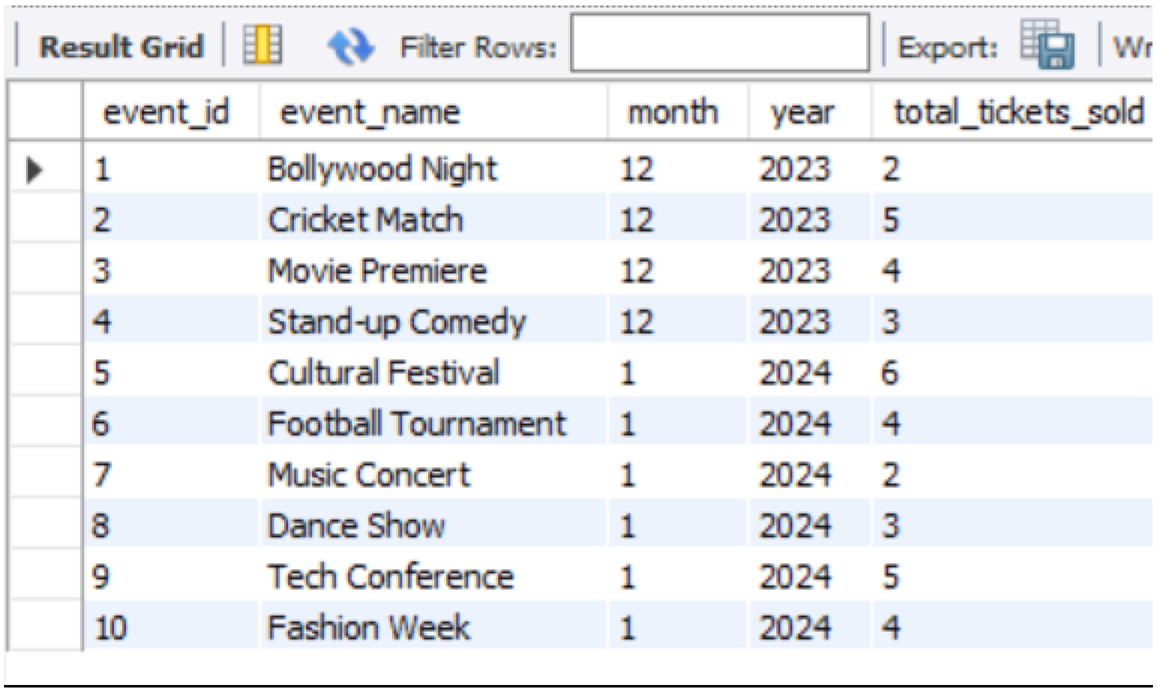
```sql
-- 6
SELECT customer_id, SUM(num_tickets) AS total_tickets_booked
FROM booking
GROUP BY customer_id
ORDER BY total_tickets_booked DESC
LIMIT 1;
```

customer_id	total_tickets_booked
5	6

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

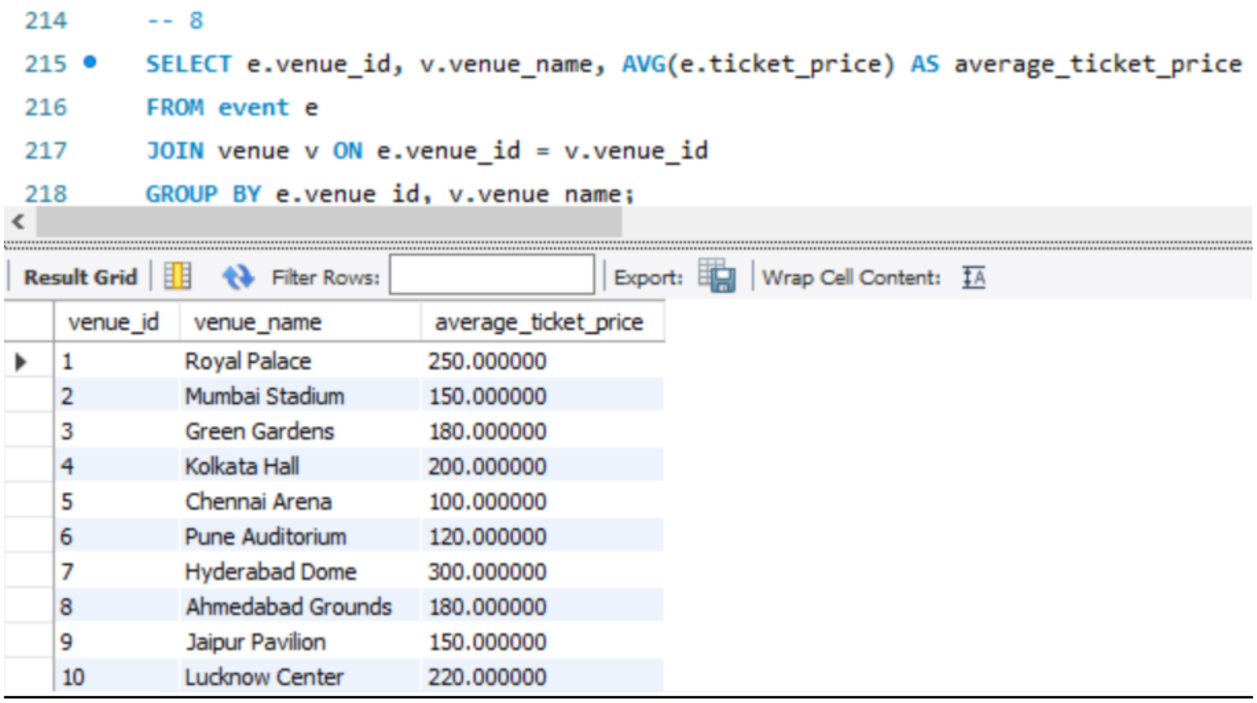


```sql
-- 7
SELECT e.event_id, e.event_name,
    EXTRACT(MONTH FROM b.booking_date) AS month,
    EXTRACT(YEAR FROM b.booking_date) AS year,
    SUM(b.num_tickets) AS total_tickets_sold
FROM event e
LEFT JOIN booking b ON e.event_id = b.event_id
GROUP BY e.event_id, e.event_name, EXTRACT(MONTH FROM b.booking_date), EXTRACT(YEAR FROM b.booking_date)
ORDER BY e.event_id, year, month;
```



	event_id	event_name	month	year	total_tickets_sold
▶	1	Bollywood Night	12	2023	2
	2	Cricket Match	12	2023	5
	3	Movie Premiere	12	2023	4
	4	Stand-up Comedy	12	2023	3
	5	Cultural Festival	1	2024	6
	6	Football Tournament	1	2024	4
	7	Music Concert	1	2024	2
	8	Dance Show	1	2024	3
	9	Tech Conference	1	2024	5
	10	Fashion Week	1	2024	4

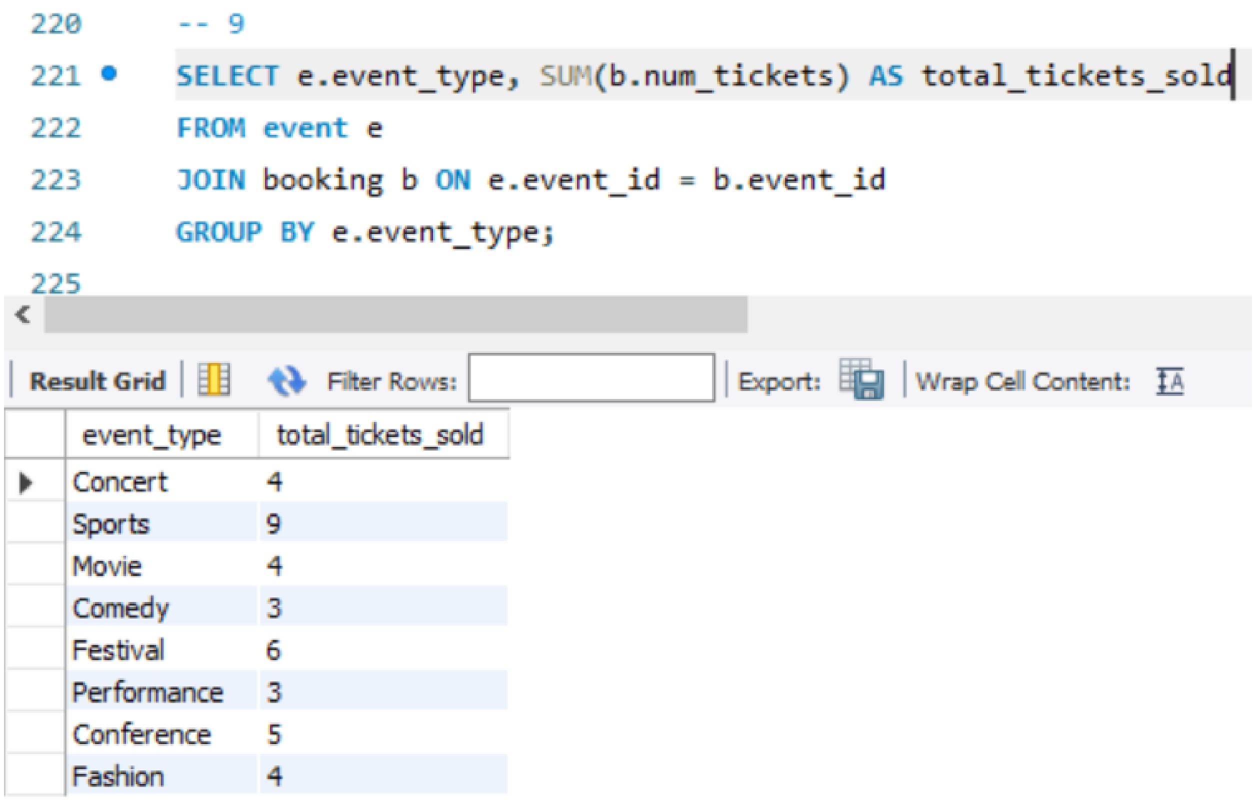
8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.



```
-- 8
SELECT e.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.venue_id, v.venue_name;
```

	venue_id	venue_name	average_ticket_price
▶	1	Royal Palace	250.000000
	2	Mumbai Stadium	150.000000
	3	Green Gardens	180.000000
	4	Kolkata Hall	200.000000
	5	Chennai Arena	100.000000
	6	Pune Auditorium	120.000000
	7	Hyderabad Dome	300.000000
	8	Ahmedabad Grounds	180.000000
	9	Jaipur Pavilion	150.000000
	10	Lucknow Center	220.000000

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

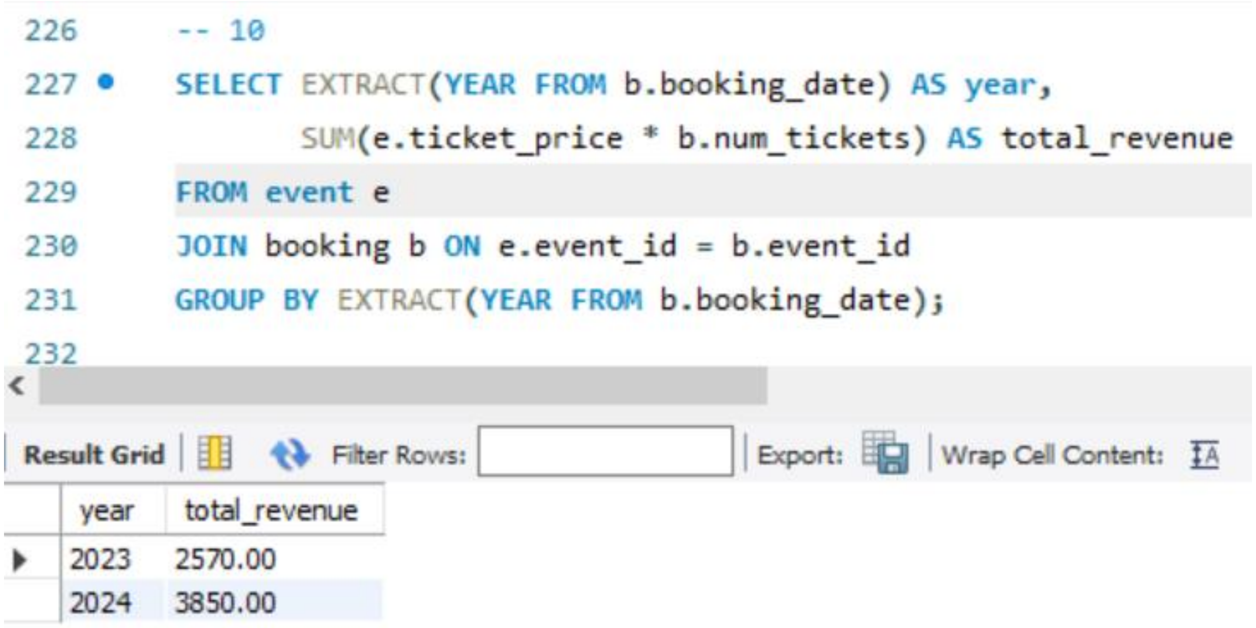


```
-- 9
SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold
FROM event e
JOIN booking b ON e.event_id = b.event_id
GROUP BY e.event_type;
```

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

event_type	total_tickets_sold
Concert	4
Sports	9
Movie	4
Comedy	3
Festival	6
Performance	3
Conference	5
Fashion	4

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

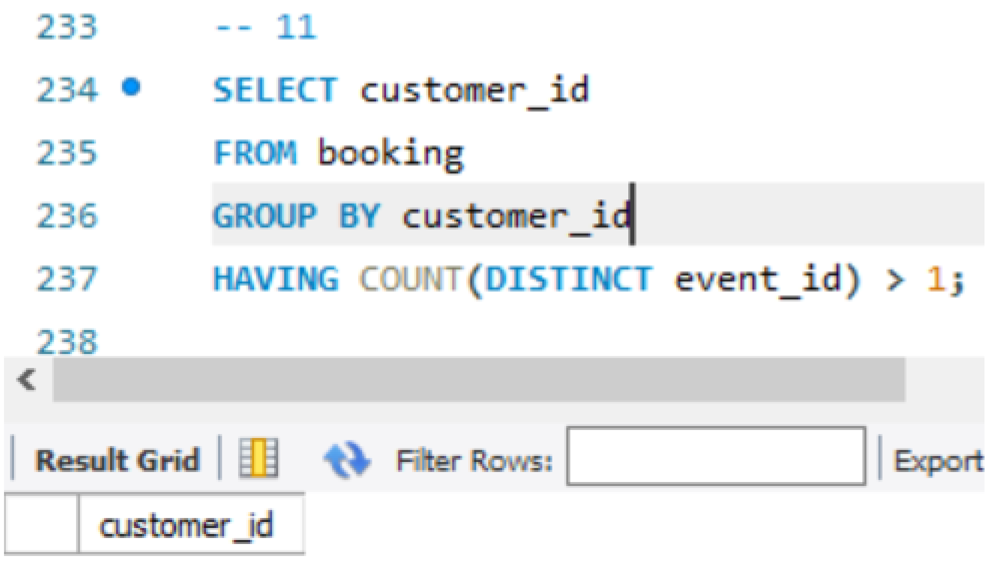


```
-- 10
SELECT EXTRACT(YEAR FROM b.booking_date) AS year,
       SUM(e.ticket_price * b.num_tickets) AS total_revenue
FROM event e
JOIN booking b ON e.event_id = b.event_id
GROUP BY EXTRACT(YEAR FROM b.booking_date);
```

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

year	total_revenue
2023	2570.00
2024	3850.00

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



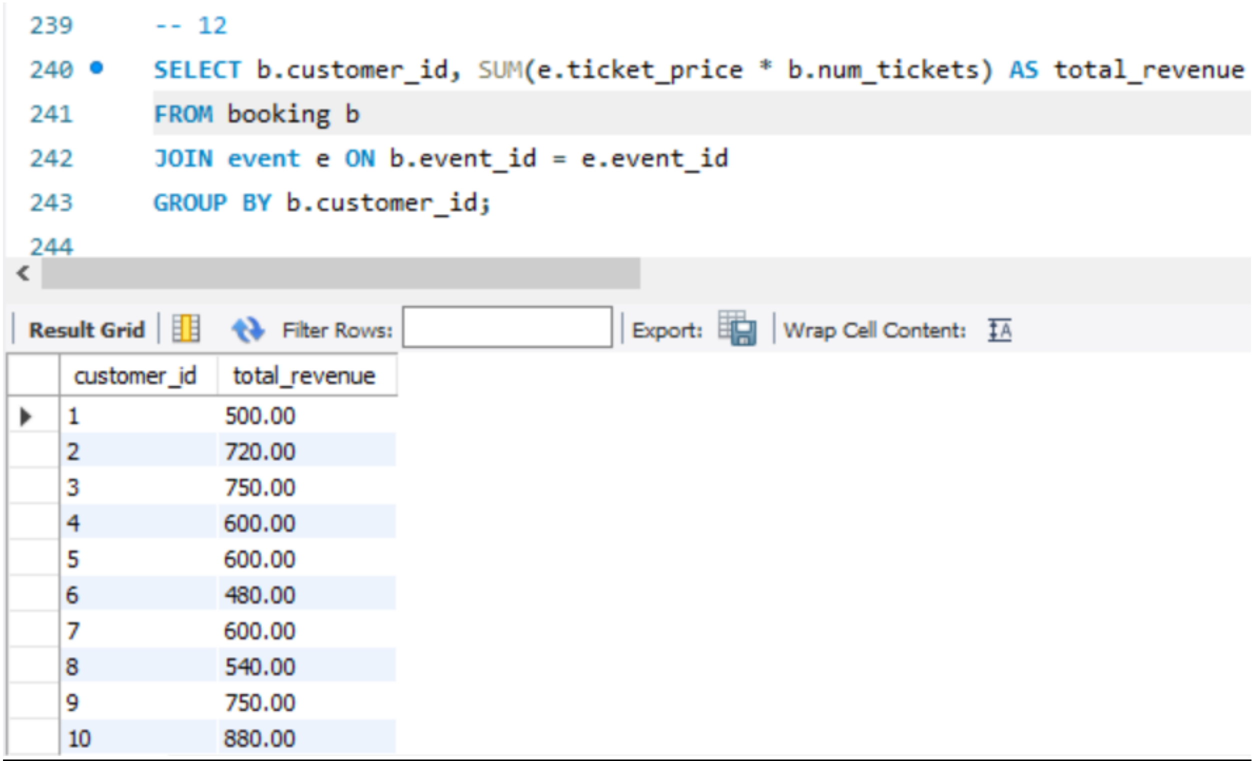
```
-- 11
SELECT customer_id
FROM booking
GROUP BY customer_id
HAVING COUNT(DISTINCT event_id) > 1;
```

Result Grid | Filter Rows: | Export

```
customer_id
```

---

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

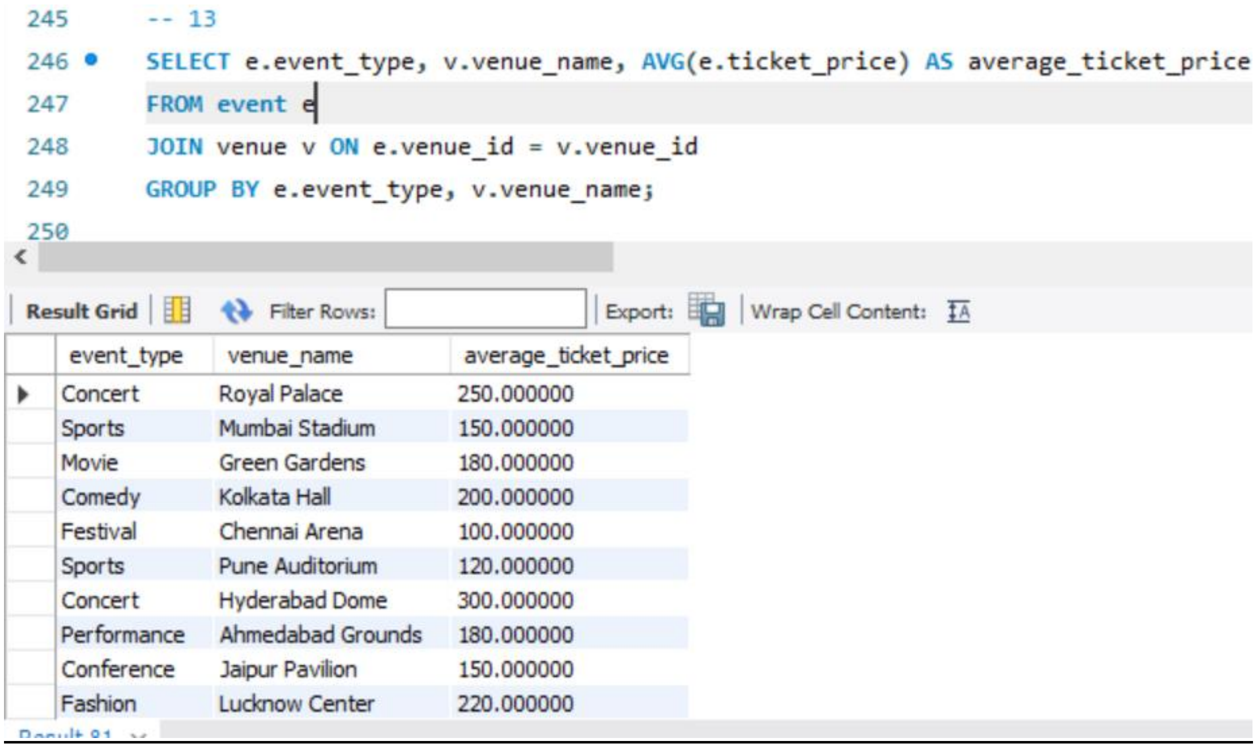


```
-- 12
SELECT b.customer_id, SUM(e.ticket_price * b.num_tickets) AS total_revenue
FROM booking b
JOIN event e ON b.event_id = e.event_id
GROUP BY b.customer_id;
```

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

customer_id	total_revenue
1	500.00
2	720.00
3	750.00
4	600.00
5	600.00
6	480.00
7	600.00
8	540.00
9	750.00
10	880.00

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



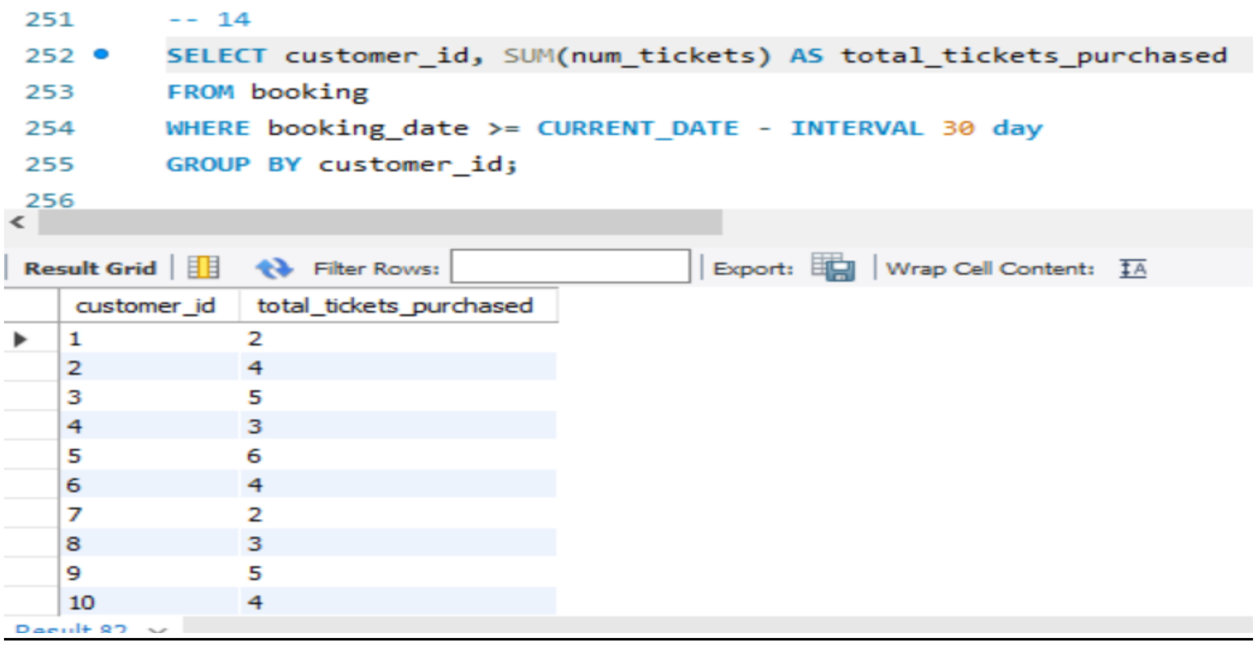
```
-- 13
SELECT e.event_type, v.venue_name, AVG(e.ticket_price) AS average_ticket_price
FROM event e
JOIN venue v ON e.venue_id = v.venue_id
GROUP BY e.event_type, v.venue_name;
```

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

event_type	venue_name	average_ticket_price
Concert	Royal Palace	250.000000
Sports	Mumbai Stadium	150.000000
Movie	Green Gardens	180.000000
Comedy	Kolkata Hall	200.000000
Festival	Chennai Arena	100.000000
Sports	Pune Auditorium	120.000000
Concert	Hyderabad Dome	300.000000
Performance	Ahmedabad Grounds	180.000000
Conference	Jaipur Pavilion	150.000000
Fashion	Lucknow Center	220.000000

Result 81

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



```
-- 14
SELECT customer_id, SUM(num_tickets) AS total_tickets_purchased
FROM booking
WHERE booking_date >= CURRENT_DATE - INTERVAL 30 day
GROUP BY customer_id;
```

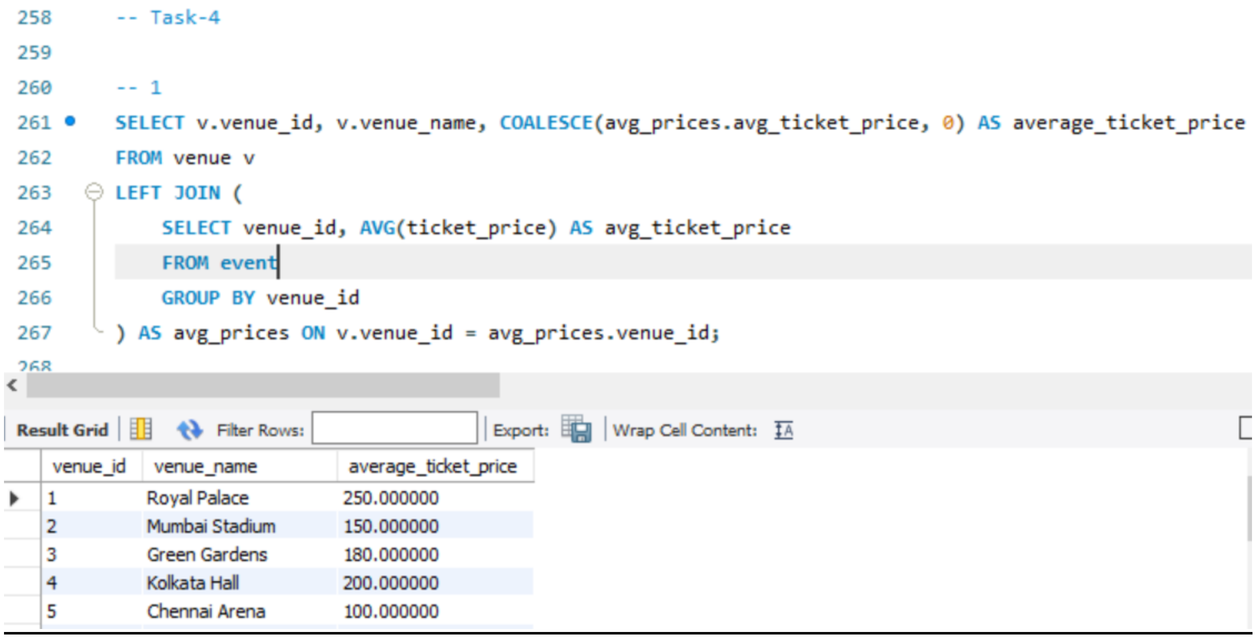
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	total_tickets_purchased
1	2
2	4
3	5
4	3
5	6
6	4
7	2
8	3
9	5
10	4

Result 82

Task-4:

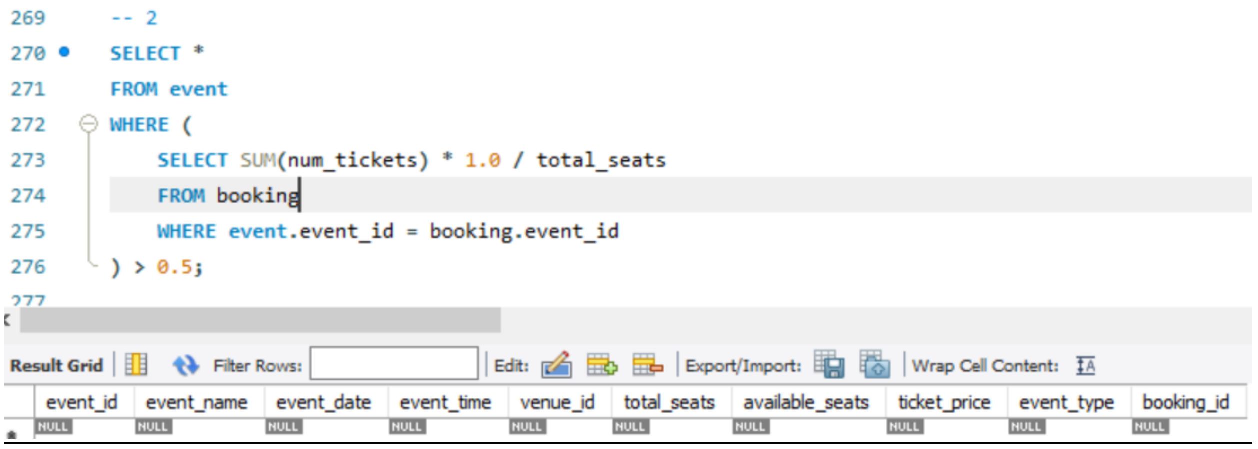
1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.



```sql
-- Task-4

-- 1
SELECT v.venue_id, v.venue_name, COALESCE(avg_prices.avg_ticket_price, 0) AS average_ticket_price
FROM venue v
LEFT JOIN (
    SELECT venue_id, AVG(ticket_price) AS avg_ticket_price
    FROM event
    GROUP BY venue_id
) AS avg_prices ON v.venue_id = avg_prices.venue_id;
```

venue_id	venue_name	average_ticket_price
1	Royal Palace	250.000000
2	Mumbai Stadium	150.000000
3	Green Gardens	180.000000
4	Kolkata Hall	200.000000
5	Chennai Arena	100.000000

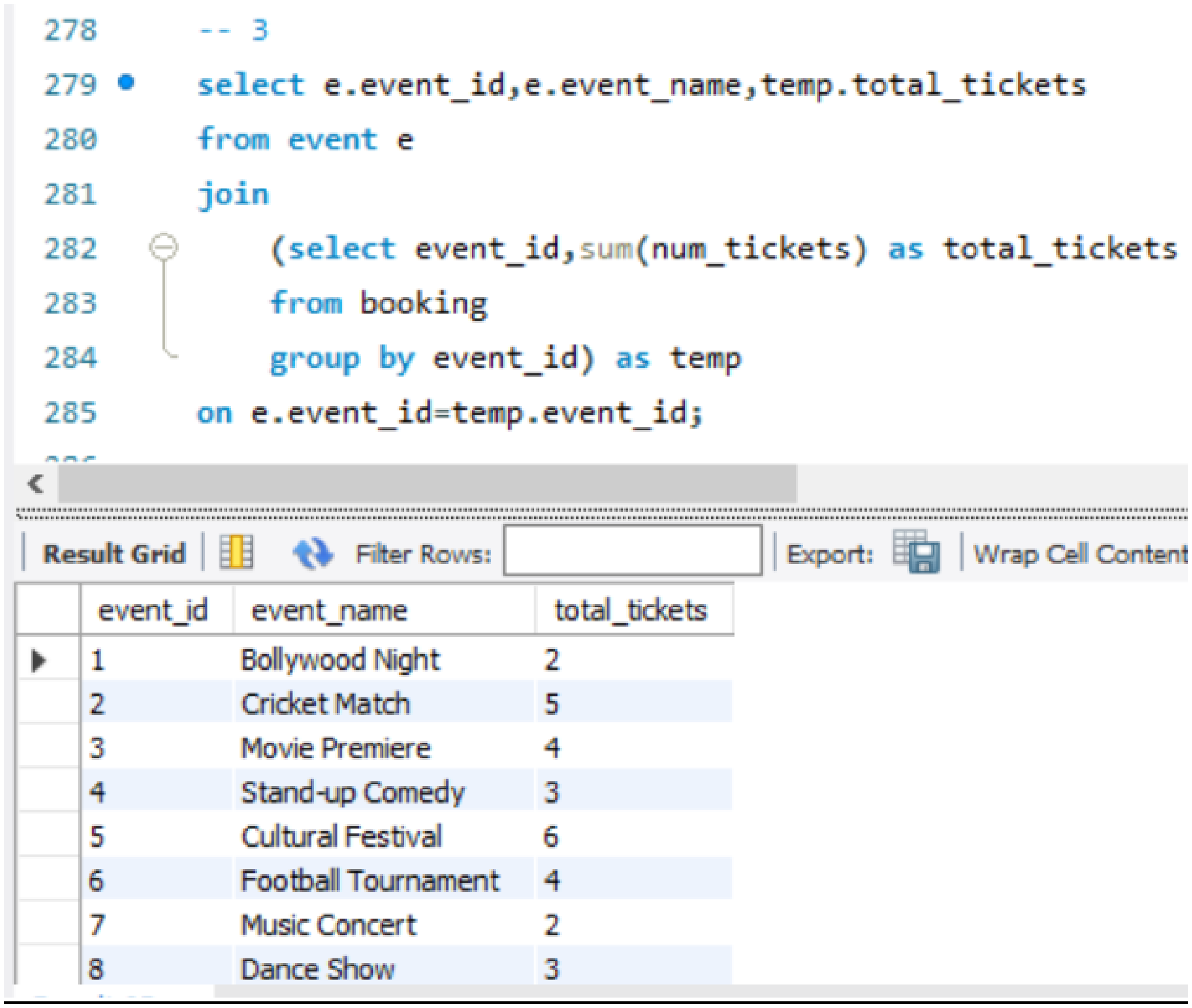
2. Find Events with More Than 50% of Tickets Sold using subquery.



```sql
-- 2
SELECT *
FROM event
WHERE (
    SELECT SUM(num_tickets) * 1.0 / total_seats
    FROM booking
    WHERE event.event_id = booking.event_id
) > 0.5;
```

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

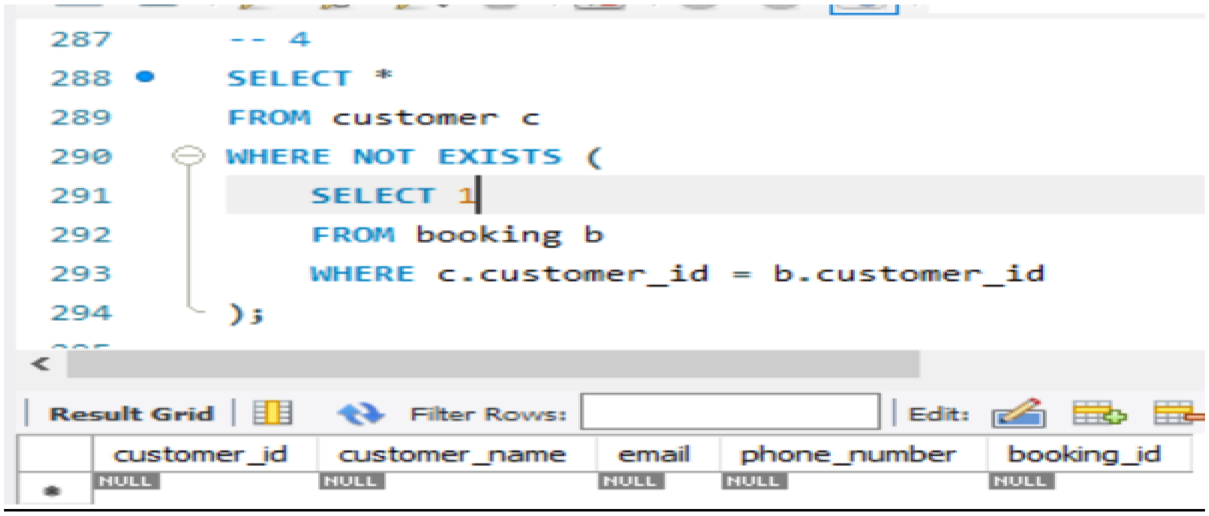
3. Calculate the Total Number of Tickets Sold for Each Event.



```
-- 3
select e.event_id,e.event_name,temp.total_tickets
from event e
join
    (select event_id,sum(num_tickets) as total_tickets
    from booking
    group by event_id) as temp
on e.event_id=temp.event_id;
```

```
event_id  event_name           total_tickets
1         Bollywood Night      2
2         Cricket Match        5
3         Movie Premiere       4
4         Stand-up Comedy      3
5         Cultural Festival    6
6         Football Tournament  4
7         Music Concert        2
8         Dance Show           3
```

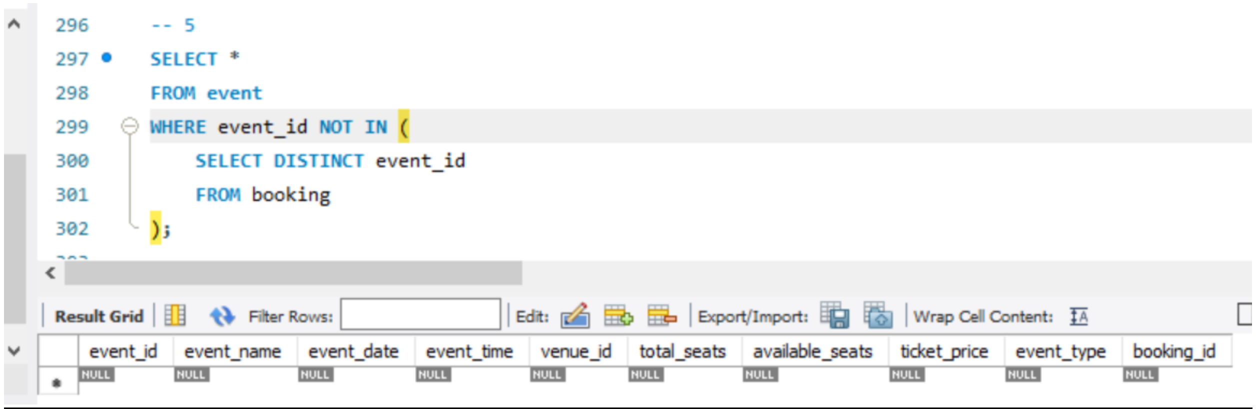
4. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.



```
-- 4
SELECT *
FROM customer c
WHERE NOT EXISTS (
    SELECT 1
    FROM booking b
    WHERE c.customer_id = b.customer_id
);
```

```
customer_id  customer_name  email  phone_number  booking_id
NULL         NULL           NULL   NULL          NULL
```

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

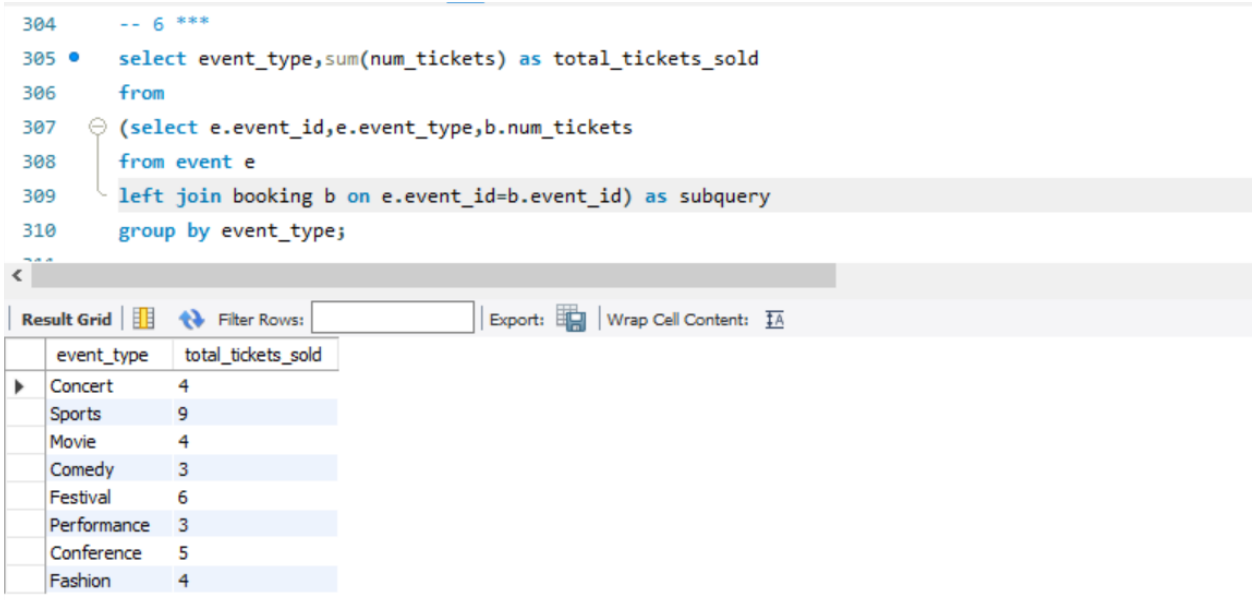


```
-- 5
SELECT *
FROM event
WHERE event_id NOT IN (
    SELECT DISTINCT event_id
    FROM booking
);
```

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
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

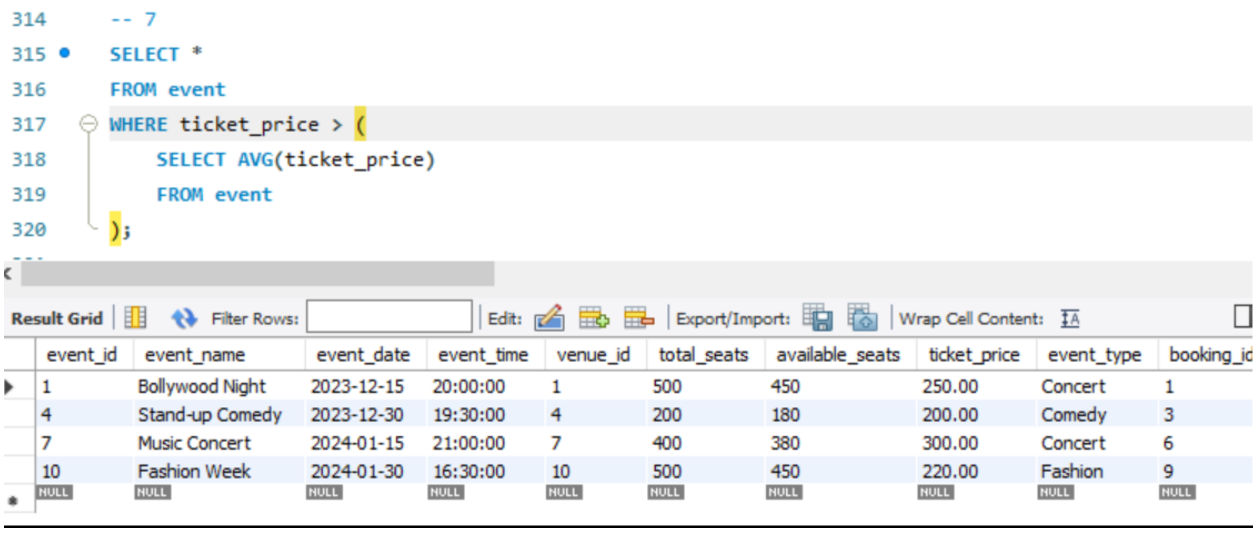


```
-- 6 ***
select event_type,sum(num_tickets) as total_tickets_sold
from
(select e.event_id,e.event_type,b.num_tickets
from event e
left join booking b on e.event_id=b.event_id) as subquery
group by event_type;
```

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

event_type	total_tickets_sold
Concert	4
Sports	9
Movie	4
Comedy	3
Festival	6
Performance	3
Conference	5
Fashion	4

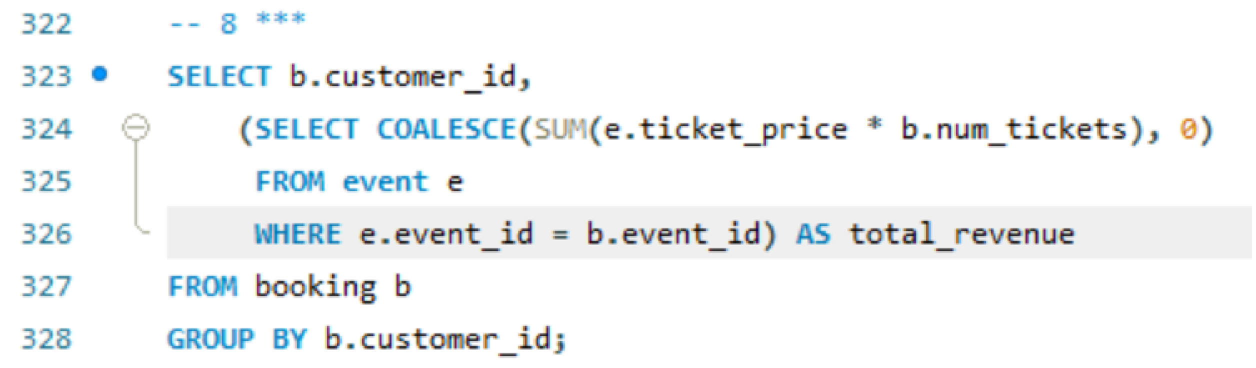
7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.



```sql
-- 7
SELECT *
FROM event
WHERE ticket_price > (
    SELECT AVG(ticket_price)
    FROM event
);
```

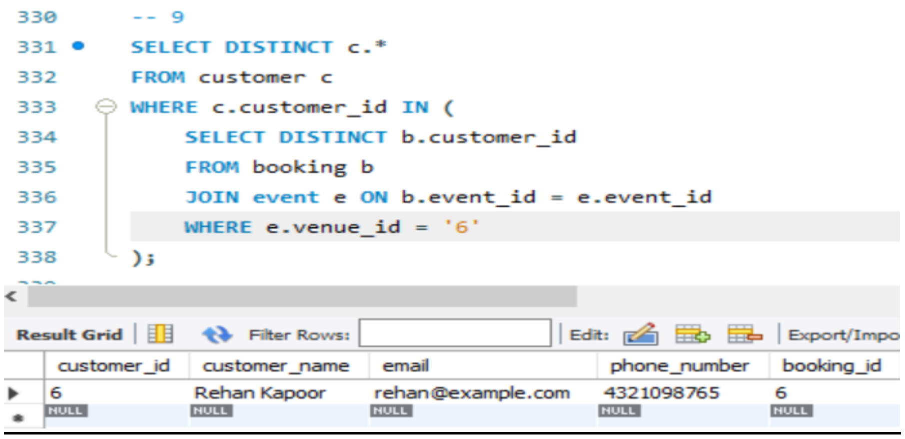
event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Bollywood Night	2023-12-15	20:00:00	1	500	450	250.00	Concert	1
4	Stand-up Comedy	2023-12-30	19:30:00	4	200	180	200.00	Comedy	3
7	Music Concert	2024-01-15	21:00:00	7	400	380	300.00	Concert	6
10	Fashion Week	2024-01-30	16:30:00	10	500	450	220.00	Fashion	9
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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



```sql
-- 8 ***
SELECT b.customer_id,
    (SELECT COALESCE(SUM(e.ticket_price * b.num_tickets), 0)
     FROM event e
     WHERE e.event_id = b.event_id) AS total_revenue
FROM booking b
GROUP BY b.customer_id;
```

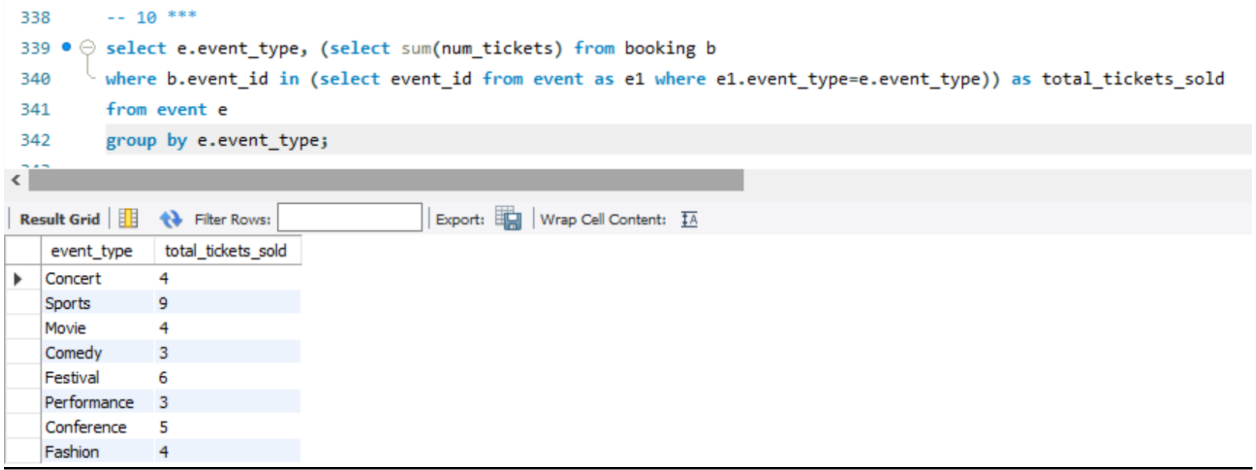
9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.



```sql
-- 9
SELECT DISTINCT c.*
FROM customer c
WHERE c.customer_id IN (
    SELECT DISTINCT b.customer_id
    FROM booking b
    JOIN event e ON b.event_id = e.event_id
    WHERE e.venue_id = '6'
);
```

customer_id	customer_name	email	phone_number	booking_id
6	Rehan Kapoor	rehan@example.com	4321098765	6
NULL	NULL	NULL	NULL	NULL

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

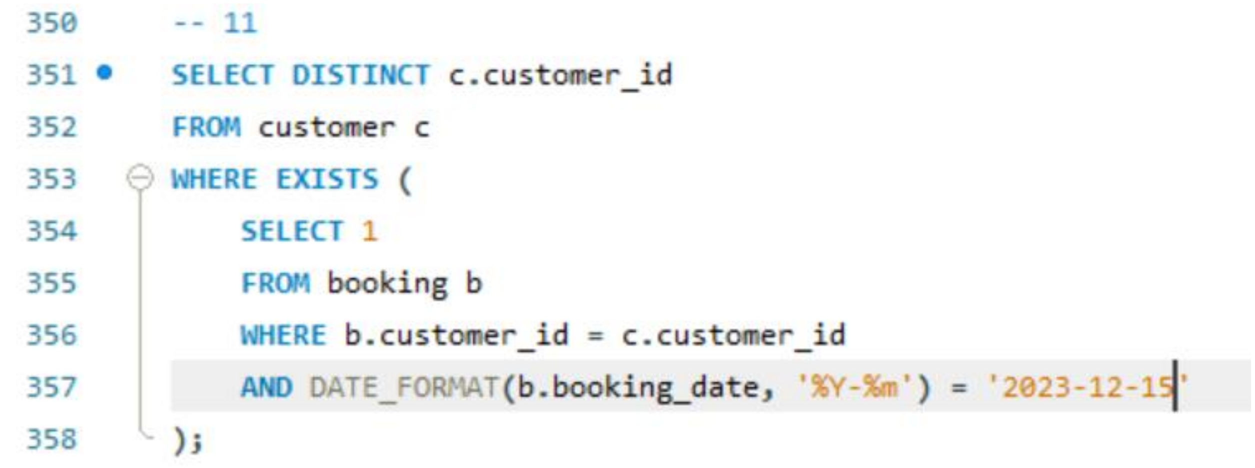


```
-- 10 ***
select e.event_type, (select sum(num_tickets) from booking b
where b.event_id in (select event_id from event as e1 where e1.event_type=e.event_type)) as total_tickets_sold
from event e
group by e.event_type;
```

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

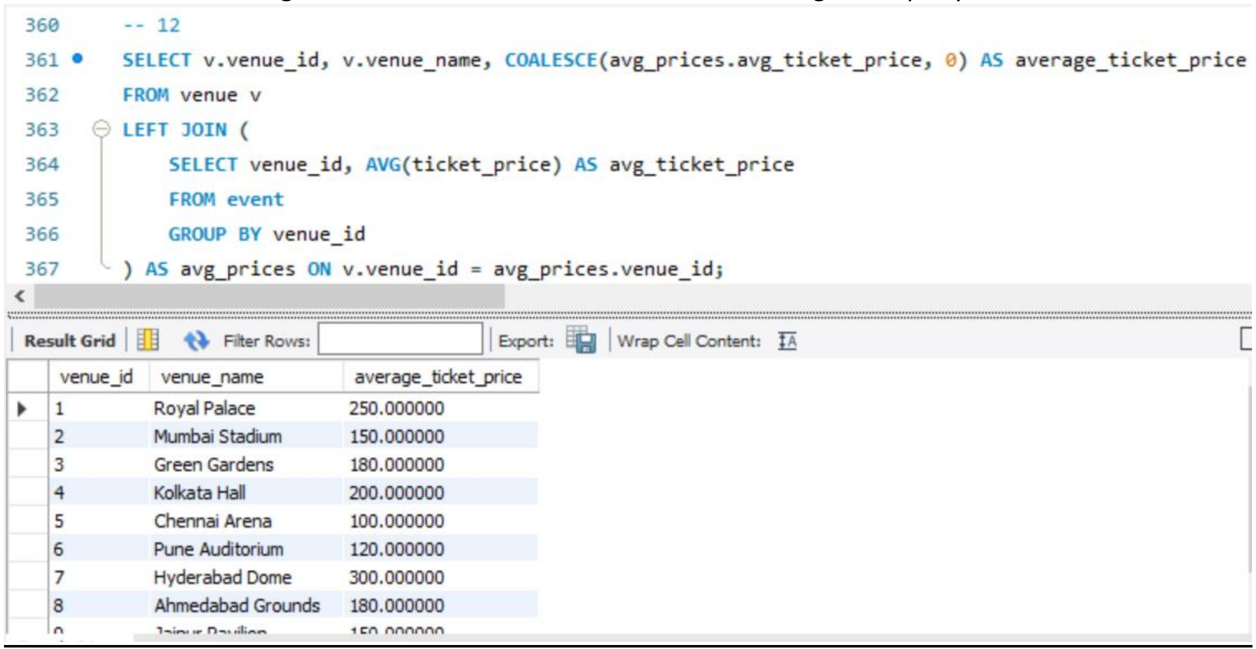
event_type	total_tickets_sold
Concert	4
Sports	9
Movie	4
Comedy	3
Festival	6
Performance	3
Conference	5
Fashion	4

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



```
-- 11
SELECT DISTINCT c.customer_id
FROM customer c
WHERE EXISTS (
    SELECT 1
    FROM booking b
    WHERE b.customer_id = c.customer_id
    AND DATE_FORMAT(b.booking_date, '%Y-%m') = '2023-12-15'
);
```

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



```sql
-- 12
SELECT v.venue_id, v.venue_name, COALESCE(avg_prices.avg_ticket_price, 0) AS average_ticket_price
FROM venue v
LEFT JOIN (
    SELECT venue_id, AVG(ticket_price) AS avg_ticket_price
    FROM event
    GROUP BY venue_id
) AS avg_prices ON v.venue_id = avg_prices.venue_id;
```

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

```
venue_id  venue_name         average_ticket_price
1         Royal Palace       250.000000
2         Mumbai Stadium     150.000000
3         Green Gardens      180.000000
4         Kolkata Hall       200.000000
5         Chennai Arena      100.000000
6         Pune Auditorium    120.000000
7         Hyderabad Dome     300.000000
8         Ahmedabad Grounds  180.000000
9         Jaipur Pavilion    150.000000
```