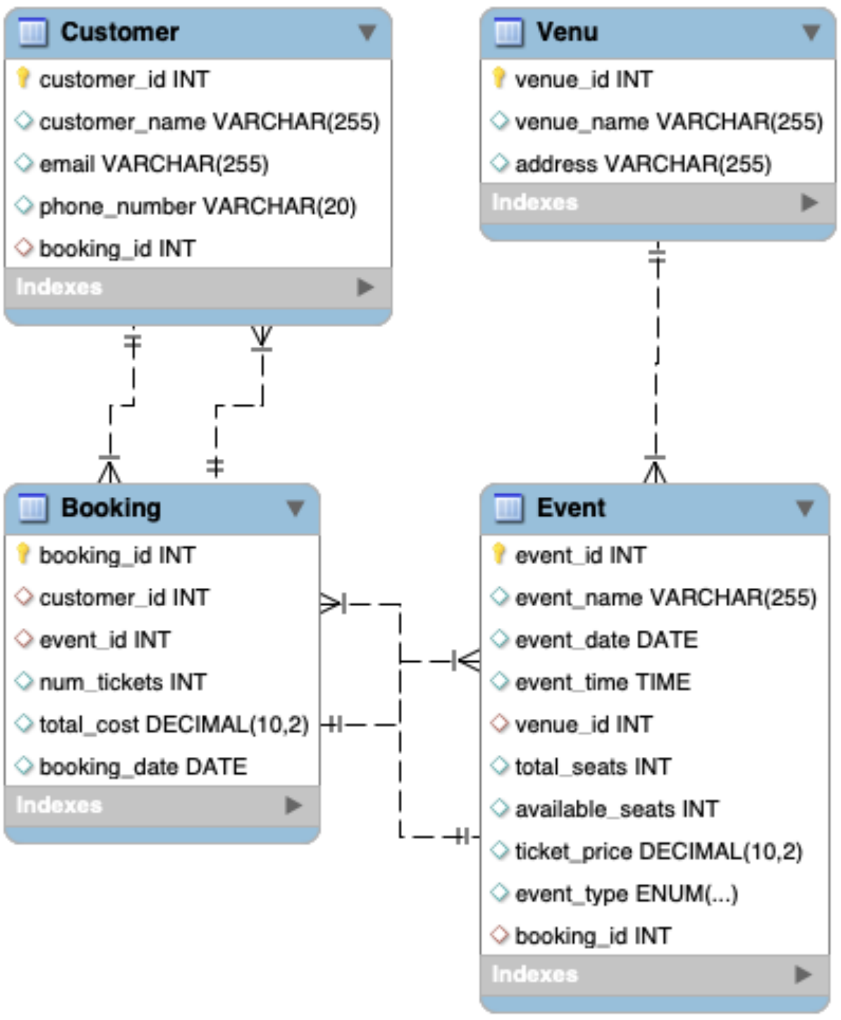
Assignment: 5

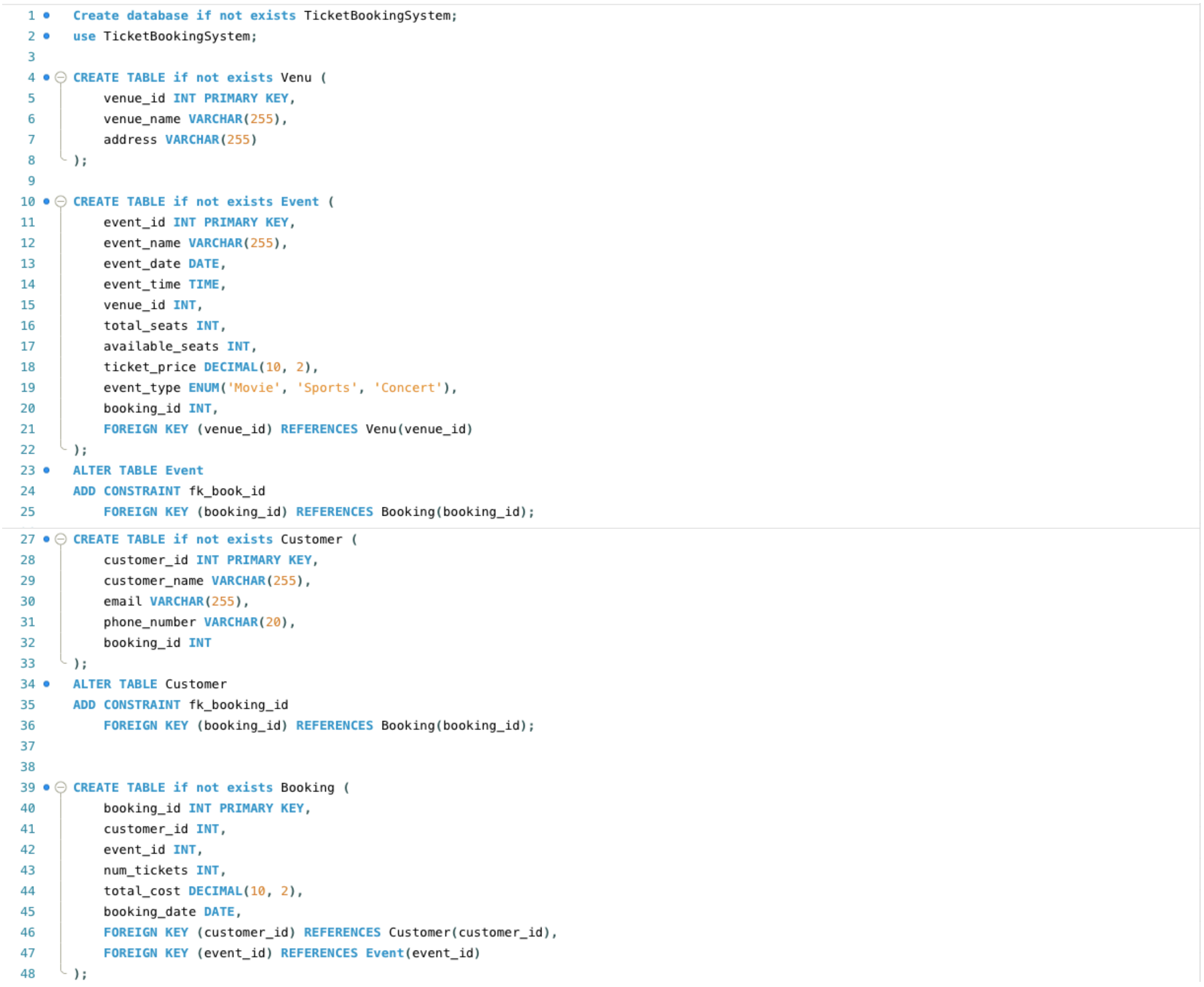
Ticket Booking System

Tasks 1: Database Design:

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



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

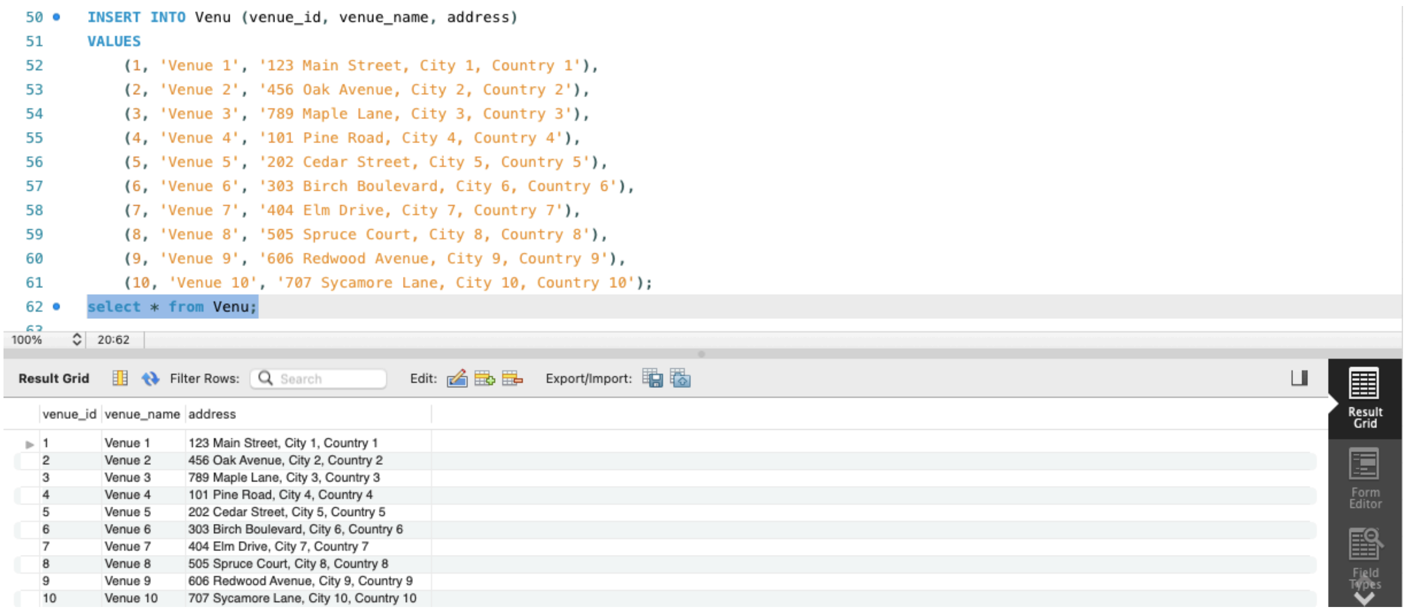


```sql
Create database if not exists TicketBookingSystem;
use TicketBookingSystem;

CREATE TABLE if not exists Venu (
    venue_id INT PRIMARY KEY,
    venue_name VARCHAR(255),
    address VARCHAR(255)
);

CREATE TABLE if not exists 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,
    FOREIGN KEY (venue_id) REFERENCES Venu(venue_id)
);
ALTER TABLE Event
ADD CONSTRAINT fk_book_id
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
```

```sql
CREATE TABLE if not exists Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(20),
    booking_id INT
);
ALTER TABLE Customer
ADD CONSTRAINT fk_booking_id
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);


CREATE TABLE if not exists 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)
);
```

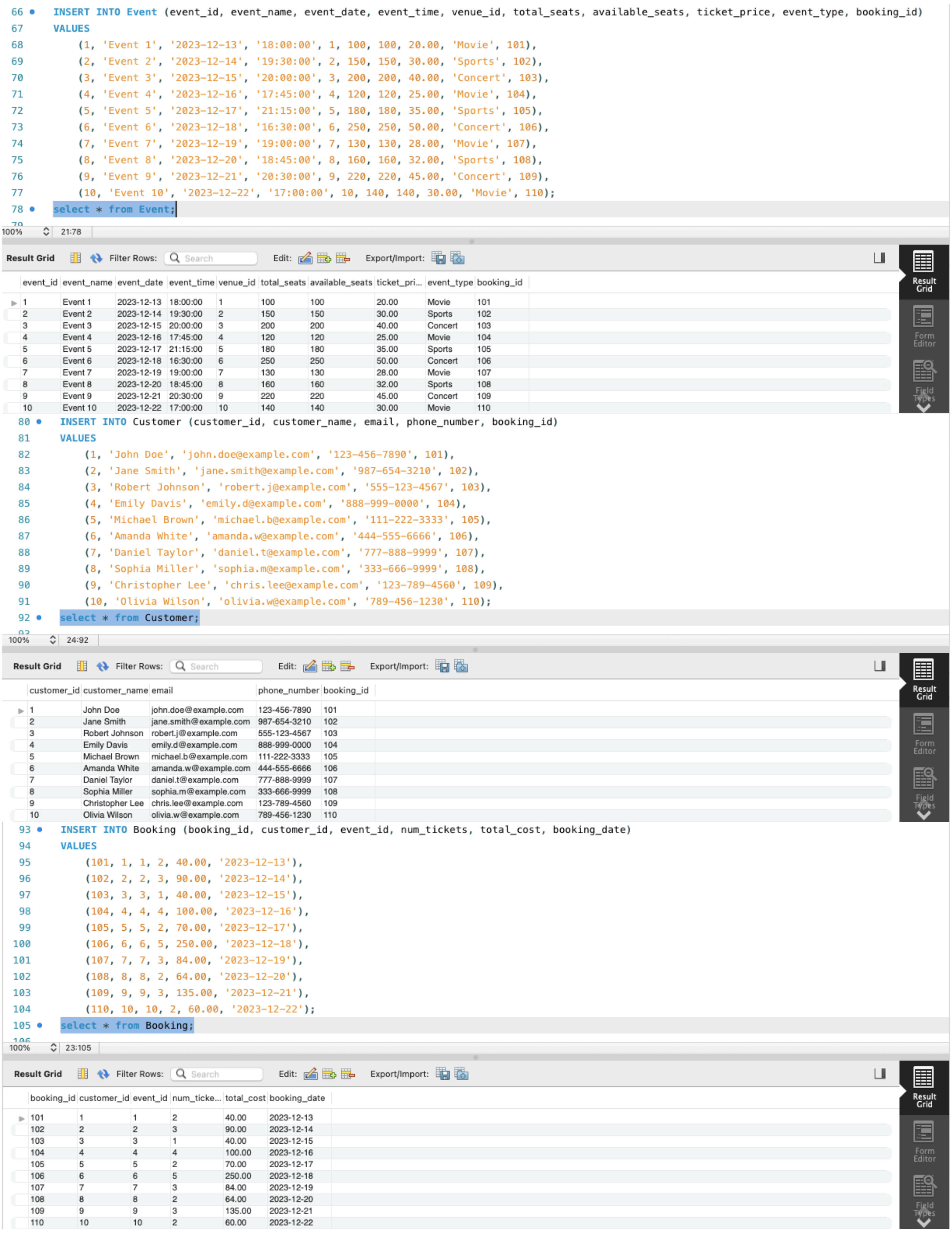
Tasks 2: Select, Where, Between, AND, LIKE:

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



```sql
INSERT INTO Venu (venue_id, venue_name, address)
VALUES
    (1, 'Venue 1', '123 Main Street, City 1, Country 1'),
    (2, 'Venue 2', '456 Oak Avenue, City 2, Country 2'),
    (3, 'Venue 3', '789 Maple Lane, City 3, Country 3'),
    (4, 'Venue 4', '101 Pine Road, City 4, Country 4'),
    (5, 'Venue 5', '202 Cedar Street, City 5, Country 5'),
    (6, 'Venue 6', '303 Birch Boulevard, City 6, Country 6'),
    (7, 'Venue 7', '404 Elm Drive, City 7, Country 7'),
    (8, 'Venue 8', '505 Spruce Court, City 8, Country 8'),
    (9, 'Venue 9', '606 Redwood Avenue, City 9, Country 9'),
    (10, 'Venue 10', '707 Sycamore Lane, City 10, Country 10');
select * from Venu;
```

venue_id	venue_name	address
1	Venue 1	123 Main Street, City 1, Country 1
2	Venue 2	456 Oak Avenue, City 2, Country 2
3	Venue 3	789 Maple Lane, City 3, Country 3
4	Venue 4	101 Pine Road, City 4, Country 4
5	Venue 5	202 Cedar Street, City 5, Country 5
6	Venue 6	303 Birch Boulevard, City 6, Country 6
7	Venue 7	404 Elm Drive, City 7, Country 7
8	Venue 8	505 Spruce Court, City 8, Country 8
9	Venue 9	606 Redwood Avenue, City 9, Country 9
10	Venue 10	707 Sycamore Lane, City 10, Country 10



```sql
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, 'Event 1', '2023-12-13', '18:00:00', 1, 100, 100, 20.00, 'Movie', 101),
    (2, 'Event 2', '2023-12-14', '19:30:00', 2, 150, 150, 30.00, 'Sports', 102),
    (3, 'Event 3', '2023-12-15', '20:00:00', 3, 200, 200, 40.00, 'Concert', 103),
    (4, 'Event 4', '2023-12-16', '17:45:00', 4, 120, 120, 25.00, 'Movie', 104),
    (5, 'Event 5', '2023-12-17', '21:15:00', 5, 180, 180, 35.00, 'Sports', 105),
    (6, 'Event 6', '2023-12-18', '16:30:00', 6, 250, 250, 50.00, 'Concert', 106),
    (7, 'Event 7', '2023-12-19', '19:00:00', 7, 130, 130, 28.00, 'Movie', 107),
    (8, 'Event 8', '2023-12-20', '18:45:00', 8, 160, 160, 32.00, 'Sports', 108),
    (9, 'Event 9', '2023-12-21', '20:30:00', 9, 220, 220, 45.00, 'Concert', 109),
    (10, 'Event 10', '2023-12-22', '17:00:00', 10, 140, 140, 30.00, 'Movie', 110);
select * from Event;
```

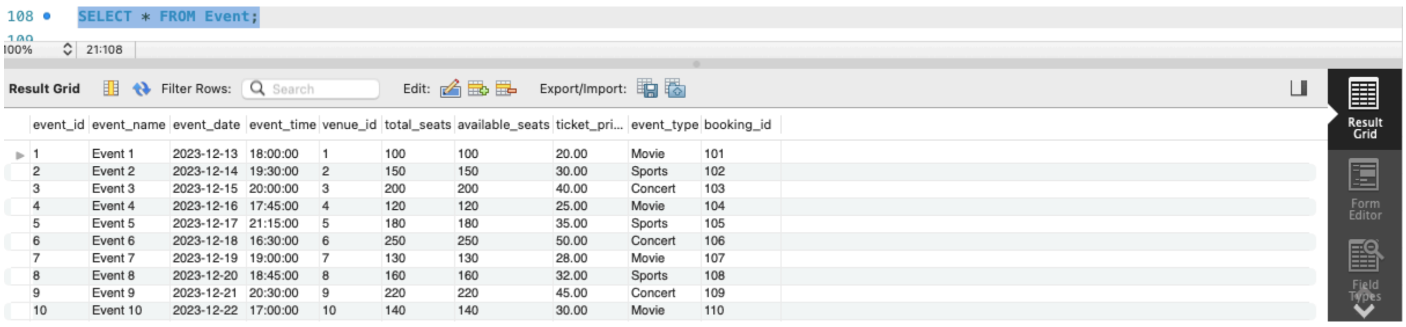
event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101
2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102
3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103
4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104
5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105
6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106
7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107
8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108
9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109
10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110

```sql
INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id)
VALUES
    (1, 'John Doe', 'john.doe@example.com', '123-456-7890', 101),
    (2, 'Jane Smith', 'jane.smith@example.com', '987-654-3210', 102),
    (3, 'Robert Johnson', 'robert.j@example.com', '555-123-4567', 103),
    (4, 'Emily Davis', 'emily.d@example.com', '888-999-0000', 104),
    (5, 'Michael Brown', 'michael.b@example.com', '111-222-3333', 105),
    (6, 'Amanda White', 'amanda.w@example.com', '444-555-6666', 106),
    (7, 'Daniel Taylor', 'daniel.t@example.com', '777-888-9999', 107),
    (8, 'Sophia Miller', 'sophia.m@example.com', '333-666-9999', 108),
    (9, 'Christopher Lee', 'chris.lee@example.com', '123-789-4560', 109),
    (10, 'Olivia Wilson', 'olivia.w@example.com', '789-456-1230', 110);
select * from Customer;
```

customer_id	customer_name	email	phone_number	booking_id
1	John Doe	john.doe@example.com	123-456-7890	101
2	Jane Smith	jane.smith@example.com	987-654-3210	102
3	Robert Johnson	robert.j@example.com	555-123-4567	103
4	Emily Davis	emily.d@example.com	888-999-0000	104
5	Michael Brown	michael.b@example.com	111-222-3333	105
6	Amanda White	amanda.w@example.com	444-555-6666	106
7	Daniel Taylor	daniel.t@example.com	777-888-9999	107
8	Sophia Miller	sophia.m@example.com	333-666-9999	108
9	Christopher Lee	chris.lee@example.com	123-789-4560	109
10	Olivia Wilson	olivia.w@example.com	789-456-1230	110

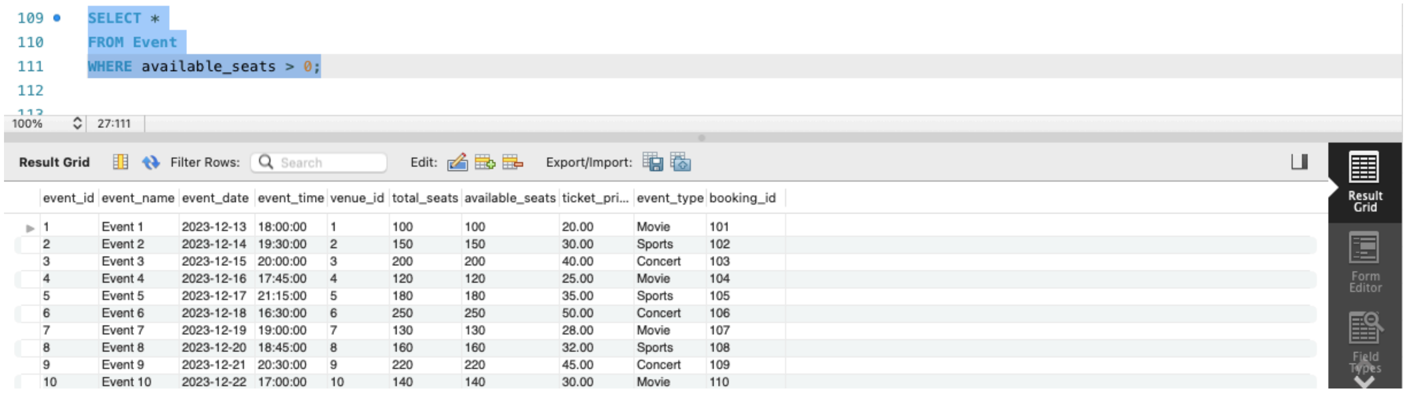
```sql
INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date)
VALUES
    (101, 1, 1, 2, 40.00, '2023-12-13'),
    (102, 2, 2, 3, 90.00, '2023-12-14'),
    (103, 3, 3, 1, 40.00, '2023-12-15'),
    (104, 4, 4, 4, 100.00, '2023-12-16'),
    (105, 5, 5, 2, 70.00, '2023-12-17'),
    (106, 6, 6, 5, 250.00, '2023-12-18'),
    (107, 7, 7, 3, 84.00, '2023-12-19'),
    (108, 8, 8, 2, 64.00, '2023-12-20'),
    (109, 9, 9, 3, 135.00, '2023-12-21'),
    (110, 10, 10, 2, 60.00, '2023-12-22');
select * from Booking;
```

booking_id	customer_id	event_id	num_ticke...	total_cost	booking_date
101	1	1	2	40.00	2023-12-13
102	2	2	3	90.00	2023-12-14
103	3	3	1	40.00	2023-12-15
104	4	4	4	100.00	2023-12-16
105	5	5	2	70.00	2023-12-17
106	6	6	5	250.00	2023-12-18
107	7	7	3	84.00	2023-12-19
108	8	8	2	64.00	2023-12-20
109	9	9	3	135.00	2023-12-21
110	10	10	2	60.00	2023-12-22


```sql
SELECT * FROM Event;
```

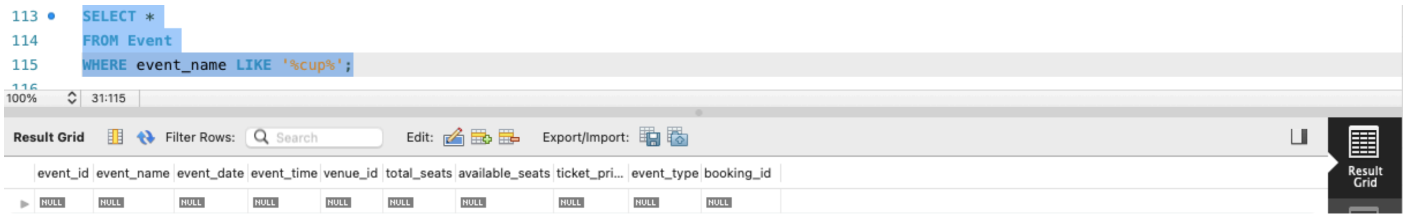
event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101
2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102
3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103
4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104
5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105
6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106
7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107
8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108
9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109
10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110

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


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

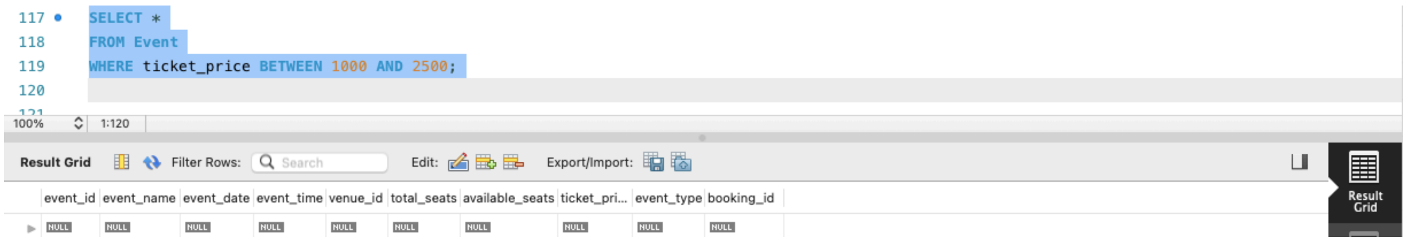
event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101
2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102
3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103
4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104
5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105
6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106
7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107
8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108
9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109
10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110

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


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

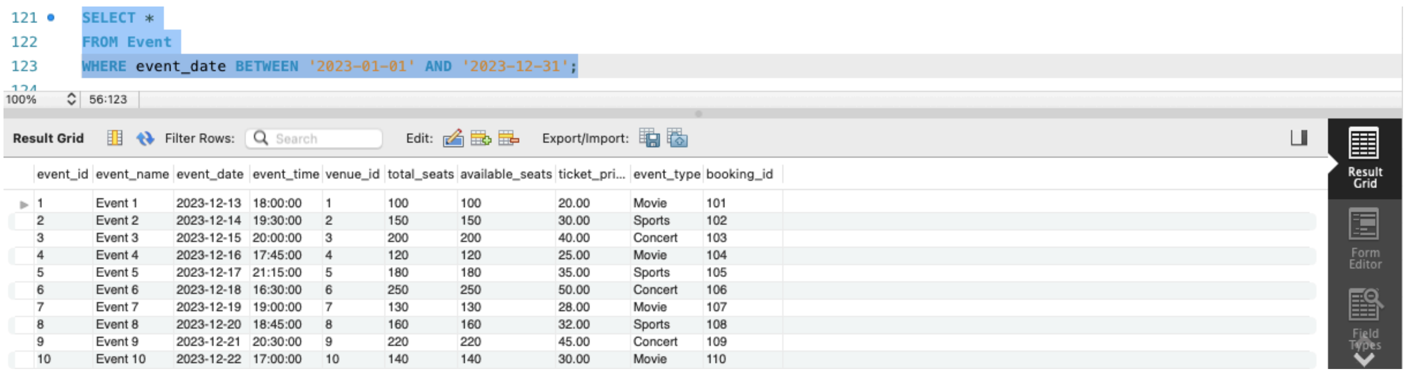
event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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


```sql
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_pri...	event_type	booking_id
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

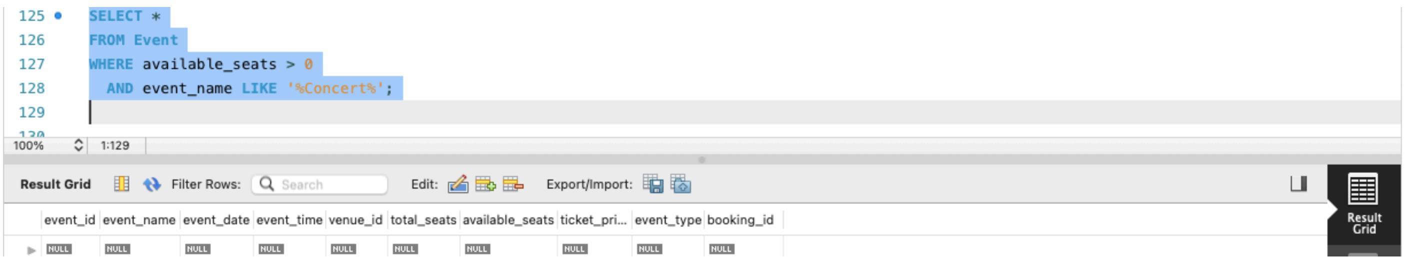
6.Write a SQL query to retrieve events with dates falling within a specific range.



```sql
SELECT *
FROM Event
WHERE event_date BETWEEN '2023-01-01' AND '2023-12-31';
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101
2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102
3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103
4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104
5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105
6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106
7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107
8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108
9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109
10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110

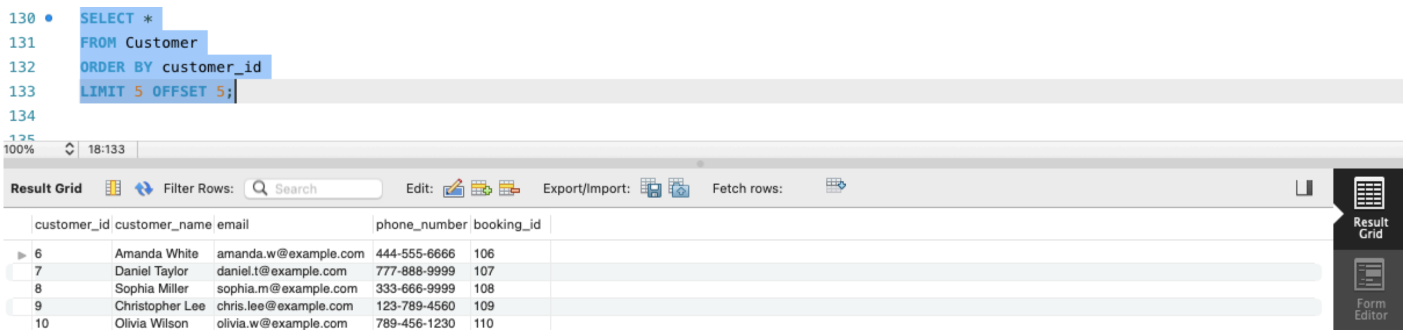
7.Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.



```sql
SELECT *
FROM Event
WHERE available_seats > 0
  AND event_name LIKE '%Concert%';
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

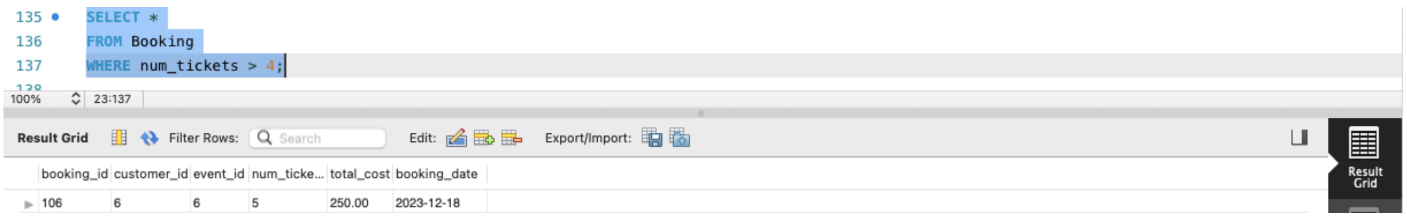
8.Write a SQL query to retrieve users in batches of 5, starting from the 6th user.



```sql
SELECT *
FROM Customer
ORDER BY customer_id
LIMIT 5 OFFSET 5;
```

customer_id	customer_name	email	phone_number	booking_id
6	Amanda White	amanda.w@example.com	444-555-6666	106
7	Daniel Taylor	daniel.t@example.com	777-888-9999	107
8	Sophia Miller	sophia.m@example.com	333-666-9999	108
9	Christopher Lee	chris.lee@example.com	123-789-4560	109
10	Olivia Wilson	olivia.w@example.com	789-456-1230	110

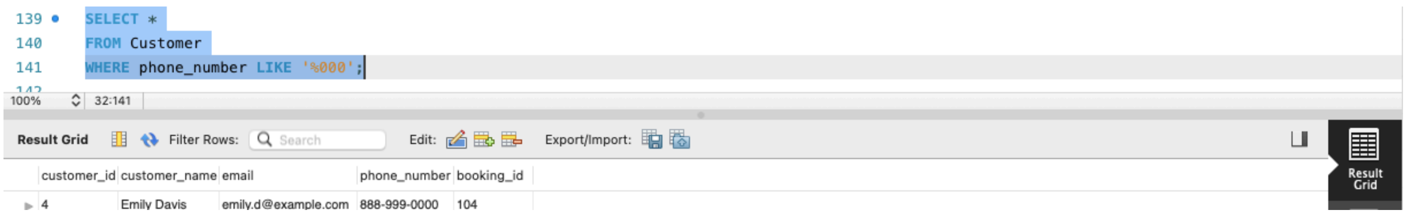
9.Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.



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

booking_id	customer_id	event_id	num_ticke...	total_cost	booking_date
106	6	6	5	250.00	2023-12-18

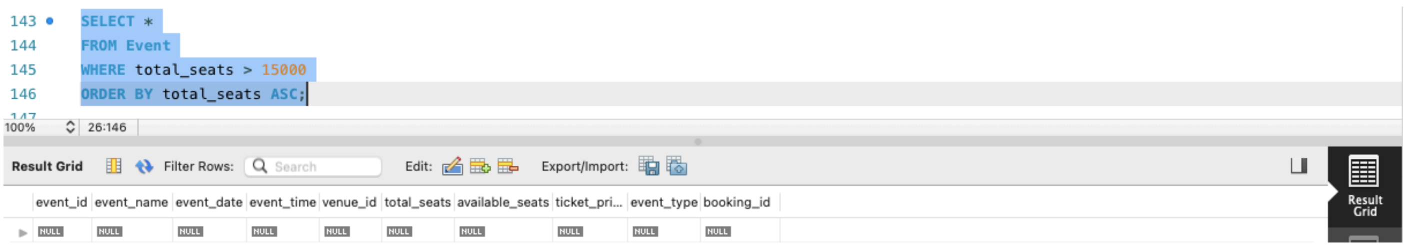
10.Write a SQL query to retrieve customer information whose phone number end with ‘000’



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

customer_id	customer_name	email	phone_number	booking_id
4	Emily Davis	emily.d@example.com	888-999-0000	104

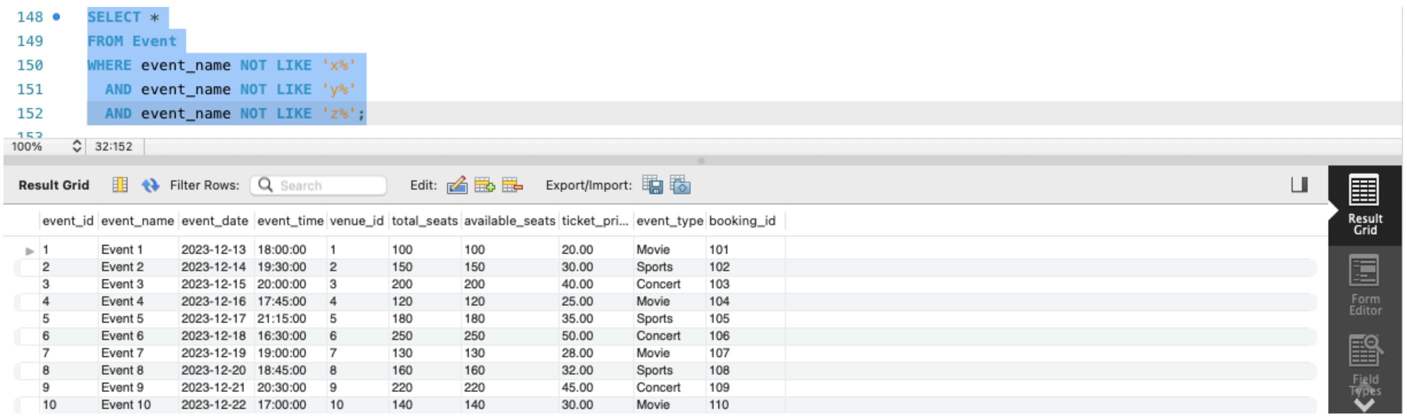
11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.



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

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	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’

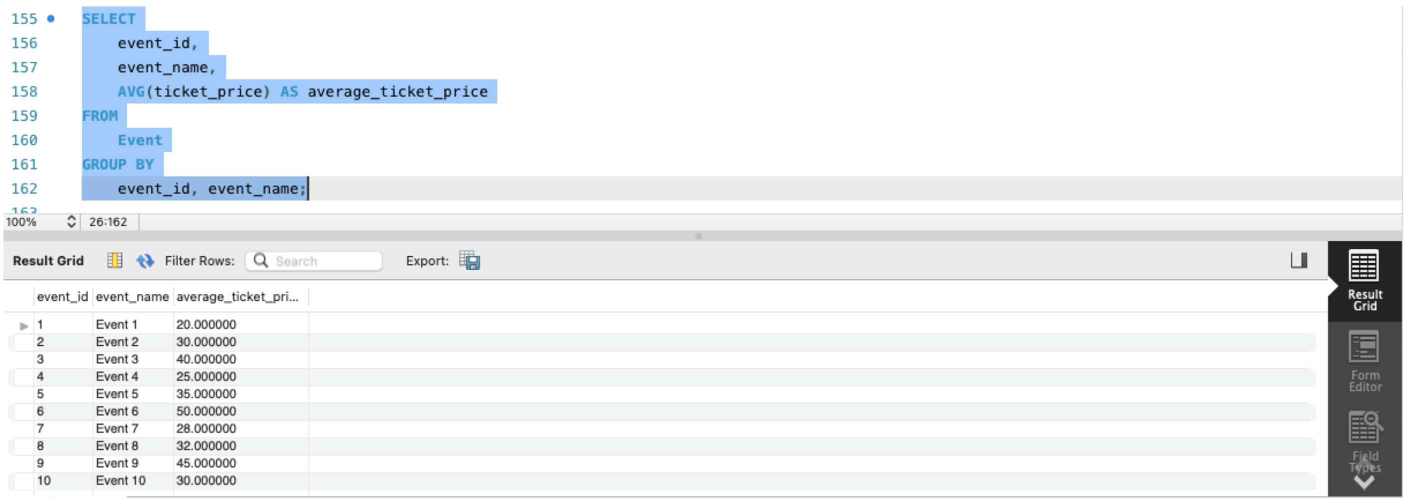


```sql
SELECT *
FROM Event
WHERE event_name NOT LIKE 'x%'
  AND event_name NOT LIKE 'y%'
  AND event_name NOT LIKE 'z%';
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101
2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102
3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103
4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104
5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105
6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106
7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107
8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108
9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109
10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110

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

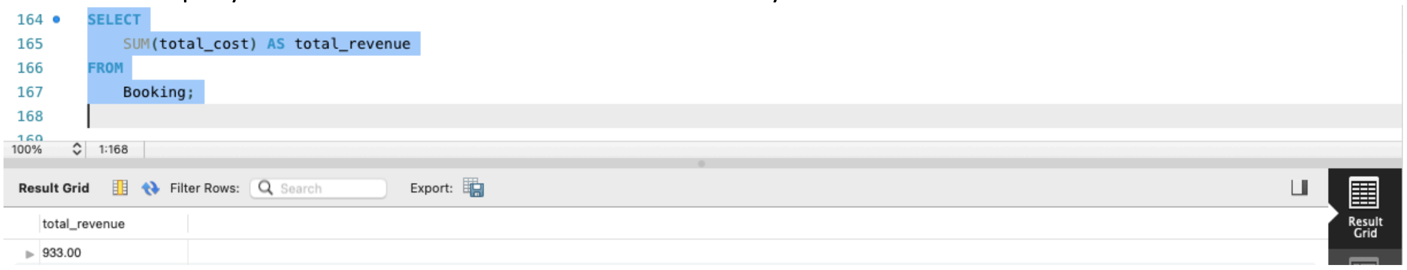
1.Write a SQL query to List Events and Their Average Ticket Prices.



```sql
SELECT
    event_id,
    event_name,
    AVG(ticket_price) AS average_ticket_price
FROM
    Event
GROUP BY
    event_id, event_name;
```

event_id	event_name	average_ticket_pri...
1	Event 1	20.000000
2	Event 2	30.000000
3	Event 3	40.000000
4	Event 4	25.000000
5	Event 5	35.000000
6	Event 6	50.000000
7	Event 7	28.000000
8	Event 8	32.000000
9	Event 9	45.000000
10	Event 10	30.000000

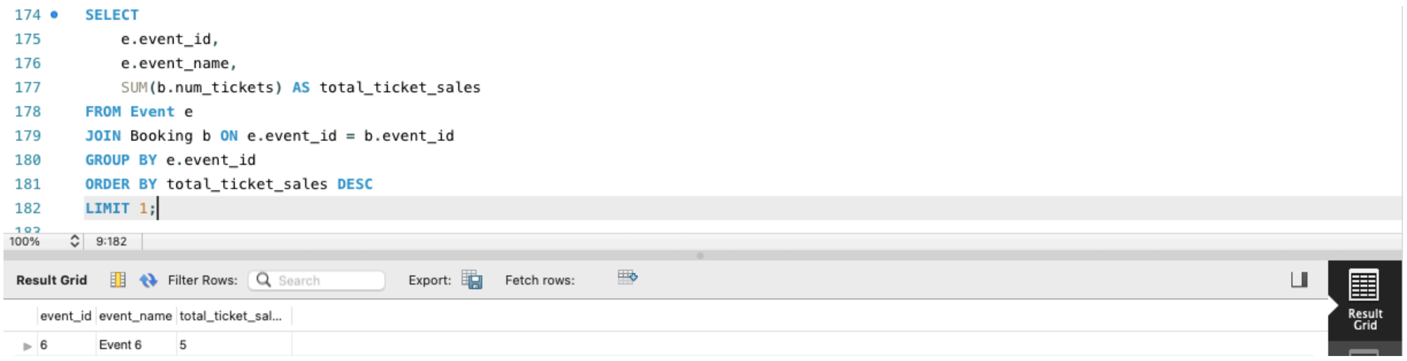
2.Write a SQL query to Calculate the Total Revenue Generated by Events.



```sql
SELECT
    SUM(total_cost) AS total_revenue
FROM
    Booking;
```

total_revenue
933.00

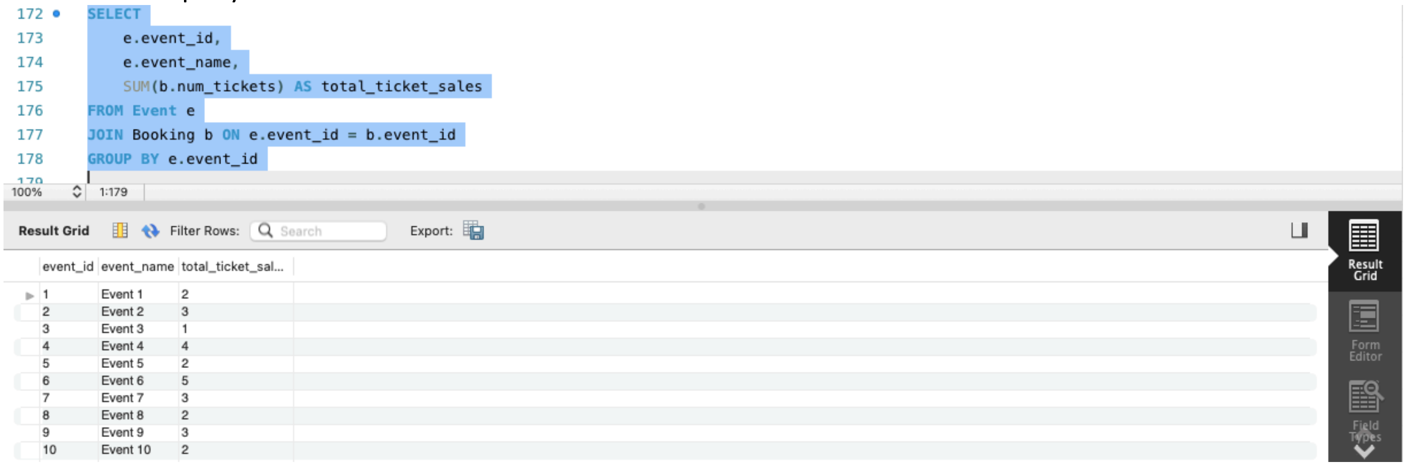
3.Write a SQL query to find the event with the highest ticket sales.



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

event_id	event_name	total_ticket_sal...
6	Event 6	5

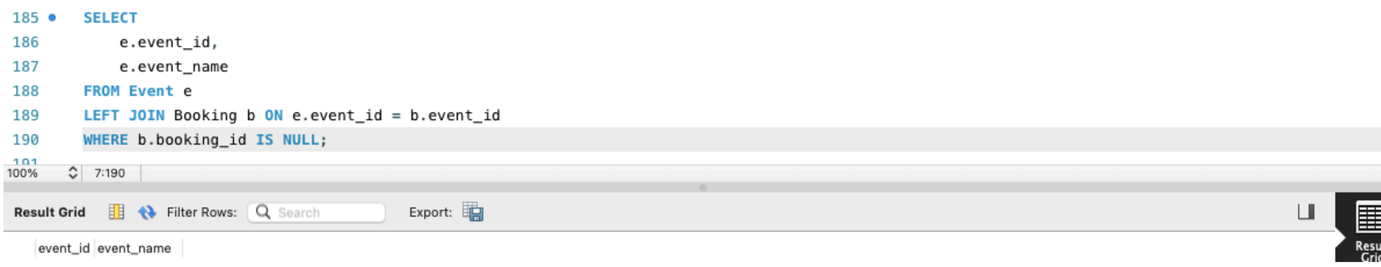
4.Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.



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

event_id	event_name	total_ticket_sal...
1	Event 1	2
2	Event 2	3
3	Event 3	1
4	Event 4	4
5	Event 5	2
6	Event 6	5
7	Event 7	3
8	Event 8	2
9	Event 9	3
10	Event 10	2

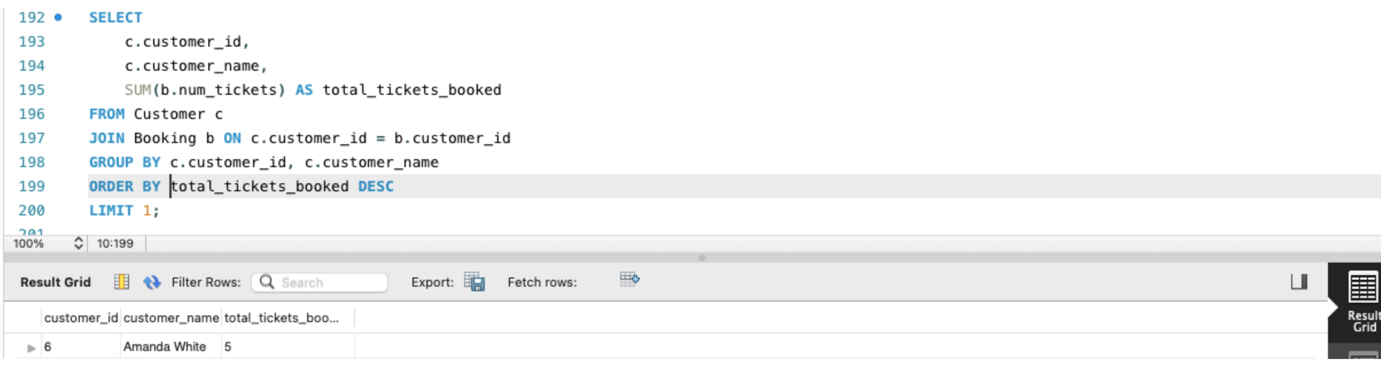
5.Write a SQL query to Find Events with No Ticket Sales.



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

```
event_id event_name
```

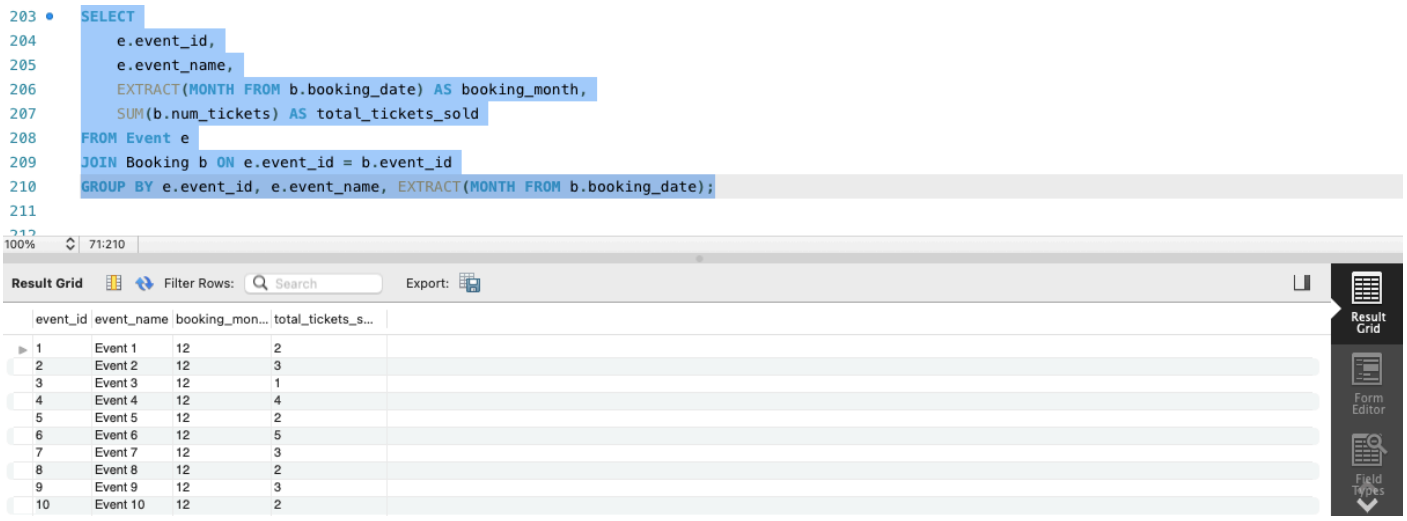
6.Write a SQL query to Find the User Who Has Booked the Most Tickets.



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

```
customer_id customer_name total_tickets_boo...
6           Amanda White  5
```

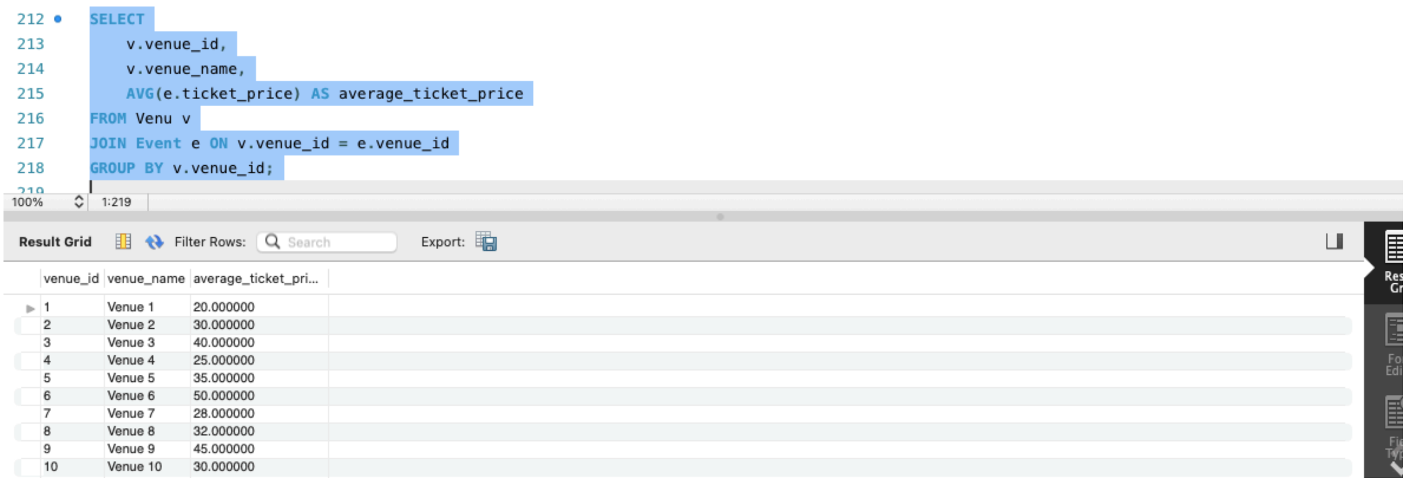
7.Write a SQL query to List Events and the total number of tickets sold for each month.



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

```
event_id event_name booking_mon... total_tickets_s...
1        Event 1    12             2
2        Event 2    12             3
3        Event 3    12             1
4        Event 4    12             4
5        Event 5    12             2
6        Event 6    12             5
7        Event 7    12             3
8        Event 8    12             2
9        Event 9    12             3
10       Event 10   12             2
```

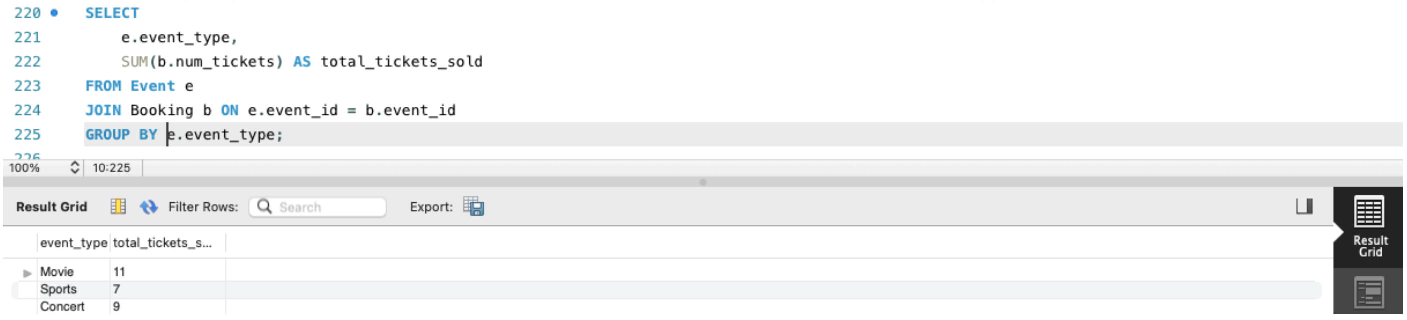
8.Write a SQL query to calculate the average Ticket Price for Events in Each Venue.



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

```
venue_id venue_name average_ticket_pri...
1        Venue 1    20.000000
2        Venue 2    30.000000
3        Venue 3    40.000000
4        Venue 4    25.000000
5        Venue 5    35.000000
6        Venue 6    50.000000
7        Venue 7    28.000000
8        Venue 8    32.000000
9        Venue 9    45.000000
10       Venue 10   30.000000
```

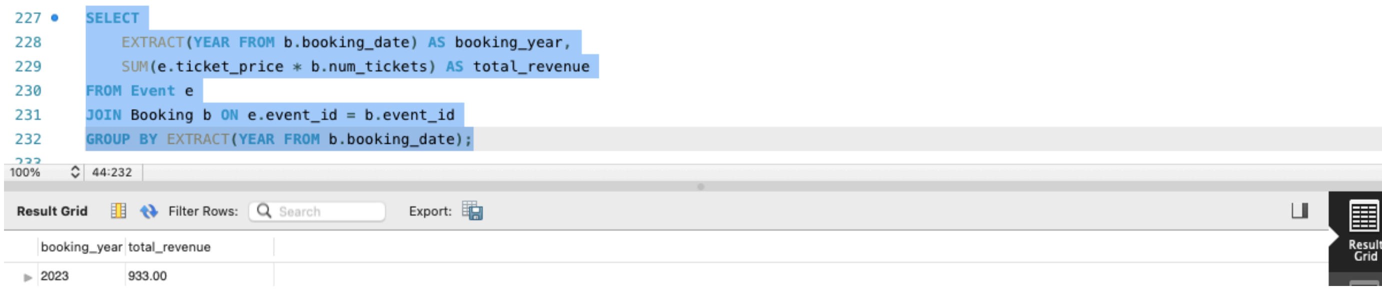
9.Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.



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

```
event_type total_tickets_s...
Movie      11
Sports     7
Concert    9
```

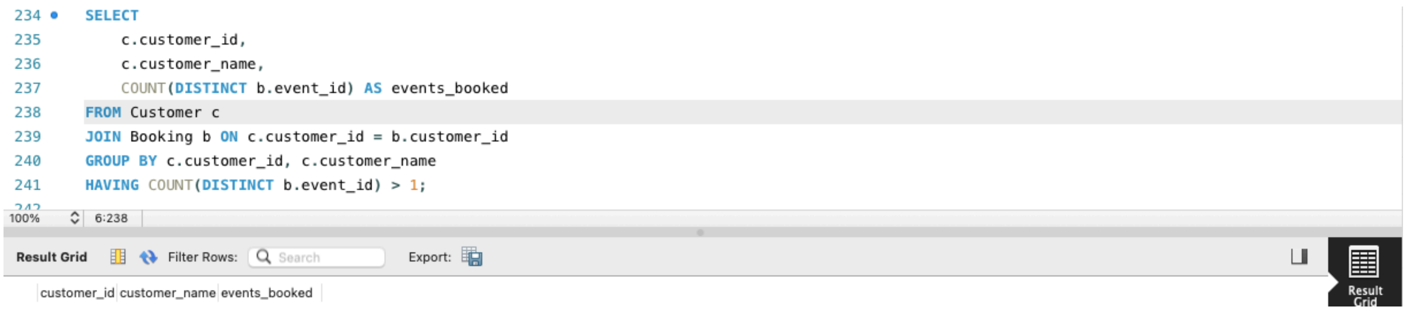
10.Write a SQL query to calculate the total Revenue Generated by Events in Each Year.



```
SELECT
    EXTRACT(YEAR FROM b.booking_date) AS booking_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);
```

```
booking_year total_revenue
2023         933.00
```

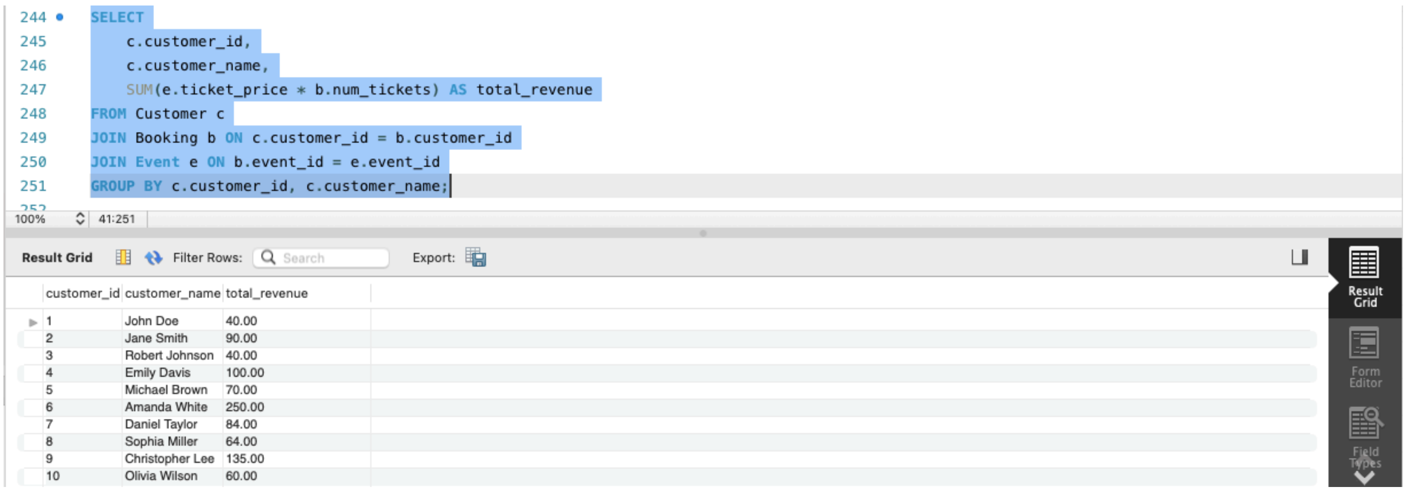
11.Write a SQL query to list users who have booked tickets for multiple events.



```
SELECT
    c.customer_id,
    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_id, c.customer_name
HAVING COUNT(DISTINCT b.event_id) > 1;
```

```
customer_id customer_name events_booked
```

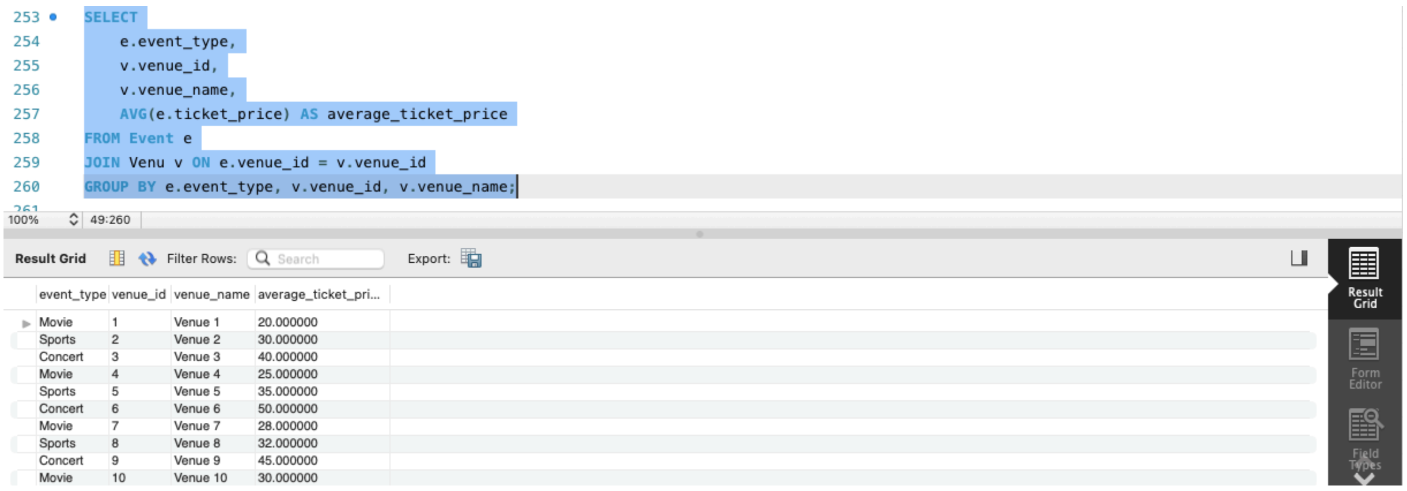
12.Write a SQL query to calculate the Total Revenue Generated by Events for Each User.



```sql
SELECT
    c.customer_id,
    c.customer_name,
    SUM(e.ticket_price * b.num_tickets) AS total_revenue
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
JOIN Event e ON b.event_id = e.event_id
GROUP BY c.customer_id, c.customer_name;
```

customer_id	customer_name	total_revenue
1	John Doe	40.00
2	Jane Smith	90.00
3	Robert Johnson	40.00
4	Emily Davis	100.00
5	Michael Brown	70.00
6	Amanda White	250.00
7	Daniel Taylor	84.00
8	Sophia Miller	64.00
9	Christopher Lee	135.00
10	Olivia Wilson	60.00

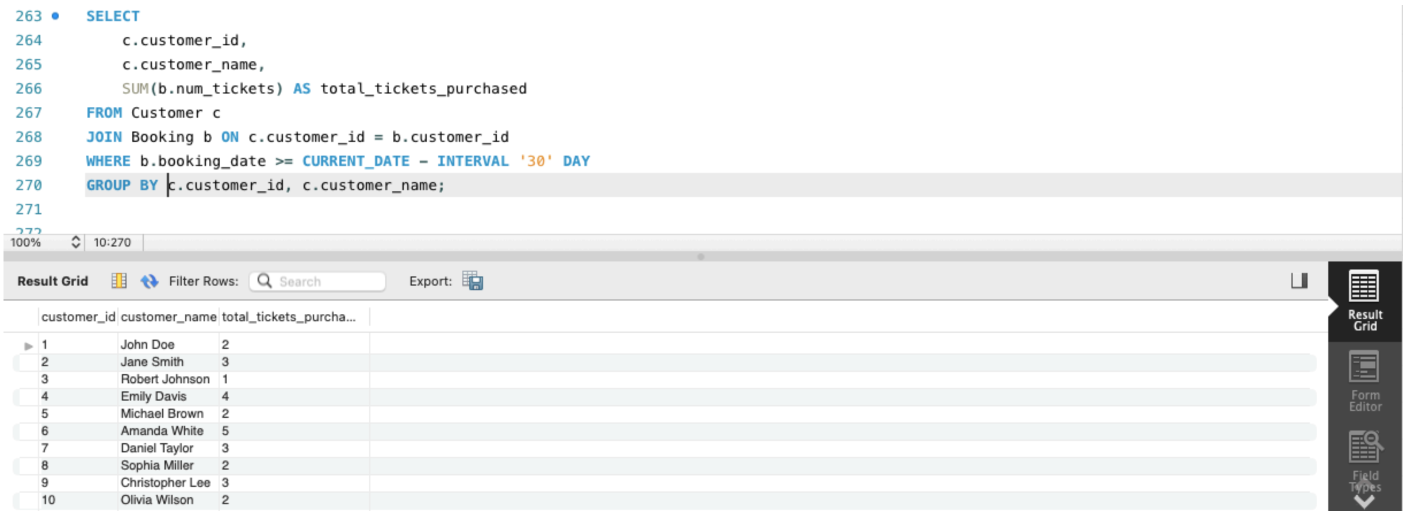
13.Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.



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

event_type	venue_id	venue_name	average_ticket_pri...
Movie	1	Venue 1	20.000000
Sports	2	Venue 2	30.000000
Concert	3	Venue 3	40.000000
Movie	4	Venue 4	25.000000
Sports	5	Venue 5	35.000000
Concert	6	Venue 6	50.000000
Movie	7	Venue 7	28.000000
Sports	8	Venue 8	32.000000
Concert	9	Venue 9	45.000000
Movie	10	Venue 10	30.000000

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

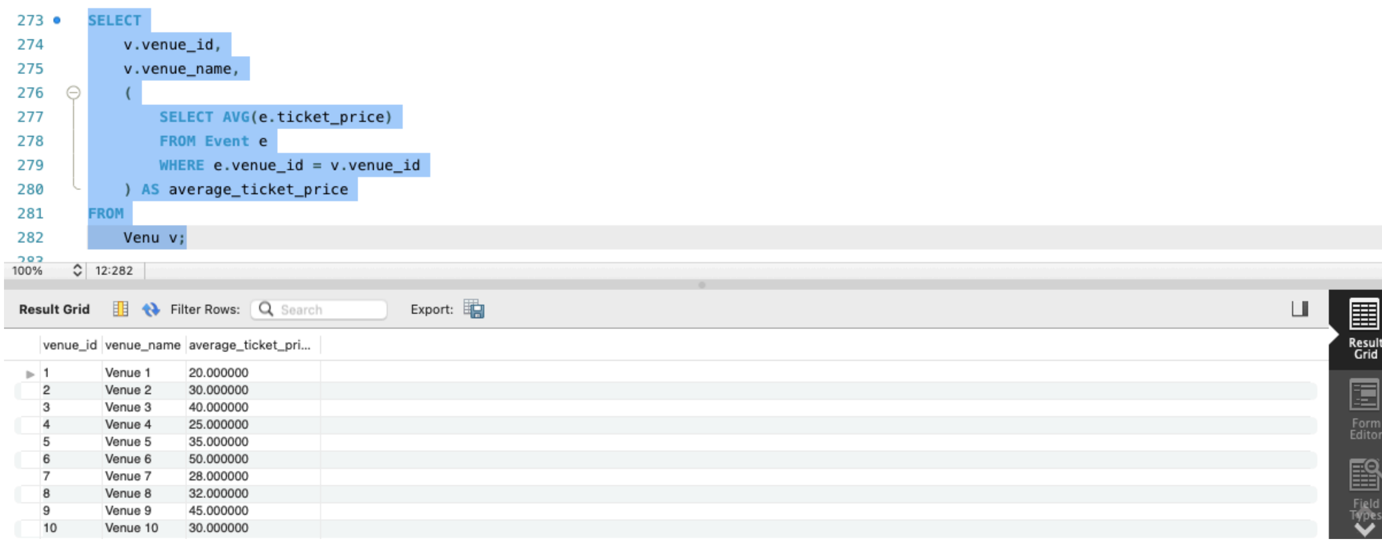


```sql
SELECT
    c.customer_id,
    c.customer_name,
    SUM(b.num_tickets) AS total_tickets_purchased
FROM Customer c
JOIN Booking b ON c.customer_id = b.customer_id
WHERE b.booking_date >= CURRENT_DATE - INTERVAL '30' DAY
GROUP BY c.customer_id, c.customer_name;
```

customer_id	customer_name	total_tickets_purcha...
1	John Doe	2
2	Jane Smith	3
3	Robert Johnson	1
4	Emily Davis	4
5	Michael Brown	2
6	Amanda White	5
7	Daniel Taylor	3
8	Sophia Miller	2
9	Christopher Lee	3
10	Olivia Wilson	2

Tasks 4: Subquery and its types

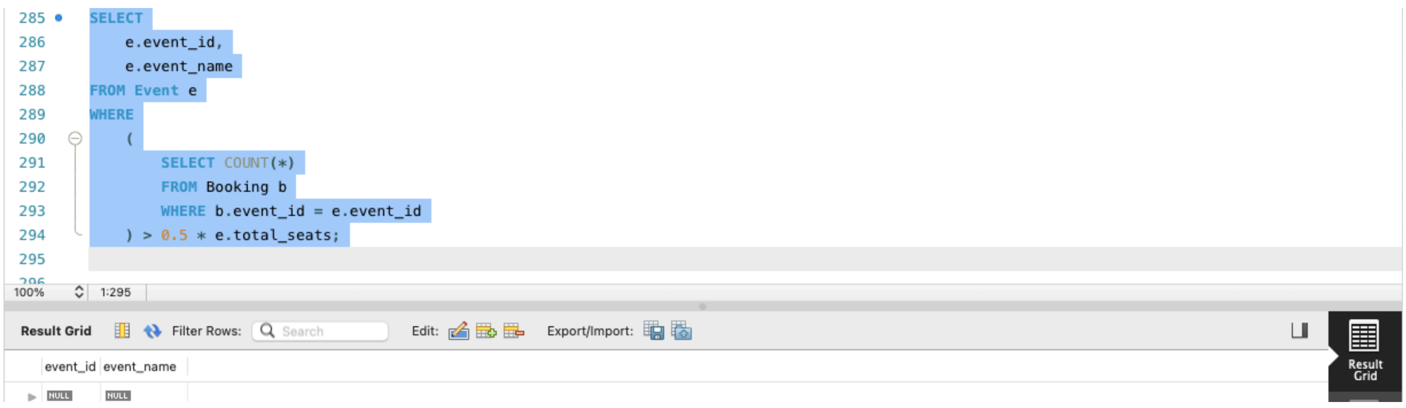
1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.



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

```
venue_id venue_name average_ticket_pri...
1        Venue 1    20.000000
2        Venue 2    30.000000
3        Venue 3    40.000000
4        Venue 4    25.000000
5        Venue 5    35.000000
6        Venue 6    50.000000
7        Venue 7    28.000000
8        Venue 8    32.000000
9        Venue 9    45.000000
10       Venue 10   30.000000
```

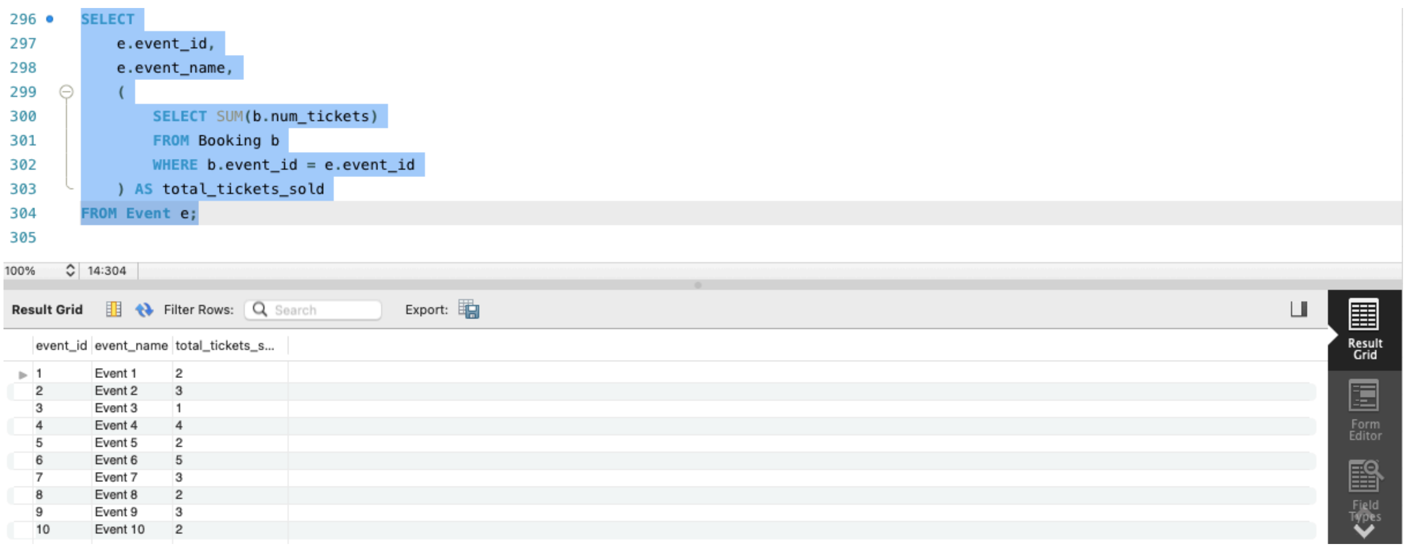
2. Find Events with More Than 50% of Tickets Sold using subquery.



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

```
event_id event_name
NULL     NULL
```

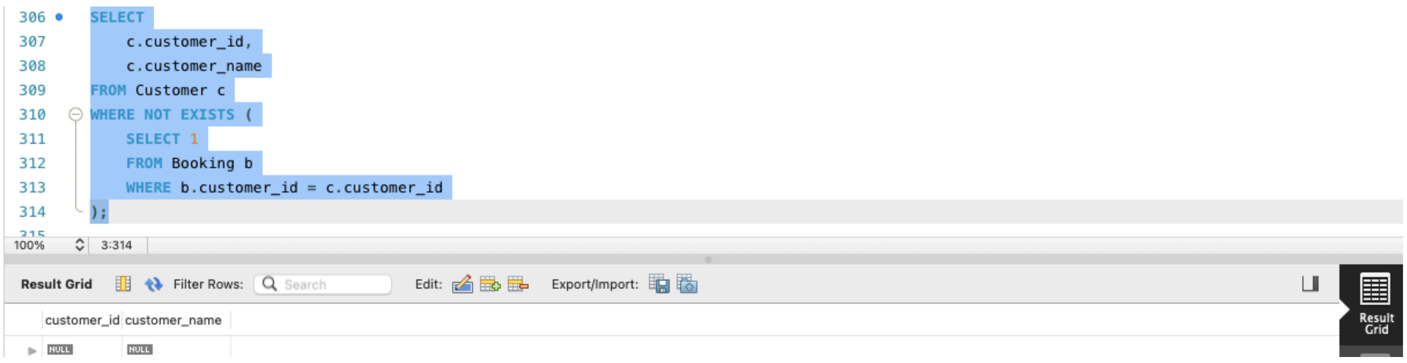
3. Calculate the Total Number of Tickets Sold for Each Event.



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

```
event_id event_name total_tickets_s...
1        Event 1    2
2        Event 2    3
3        Event 3    1
4        Event 4    4
5        Event 5    2
6        Event 6    5
7        Event 7    3
8        Event 8    2
9        Event 9    3
10       Event 10   2
```

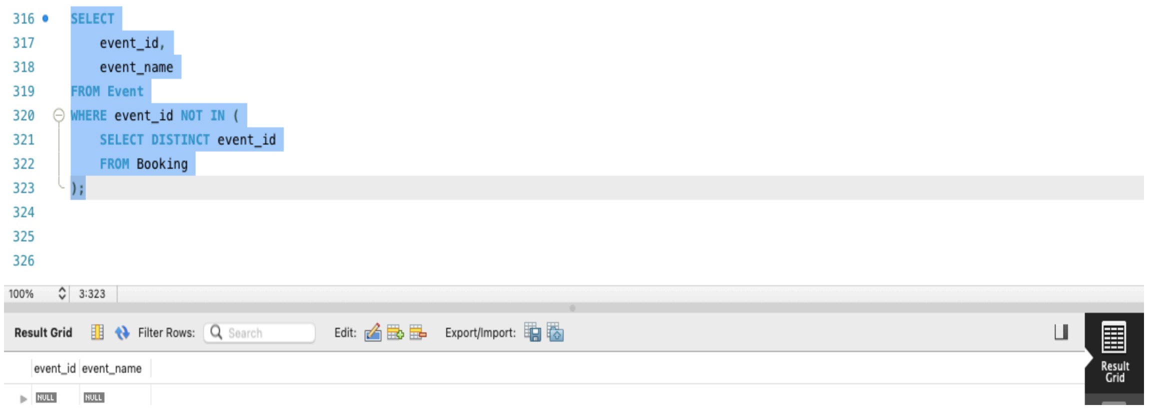
4. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.



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

```
customer_id customer_name
NULL        NULL
```

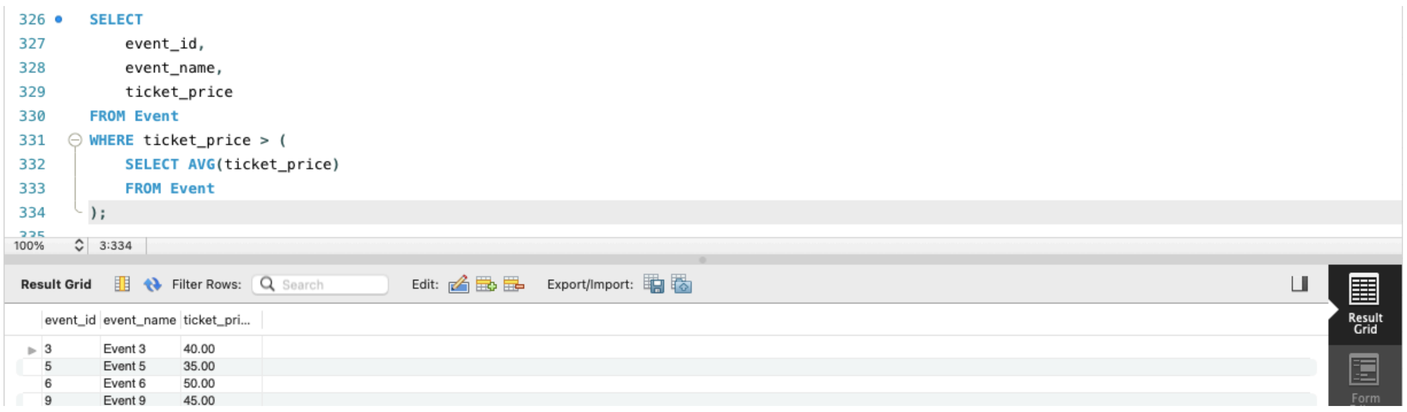
5. List Events with No Ticket Sales Using a NOT IN Subquery.



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

```
event_id event_name
NULL     NULL
```

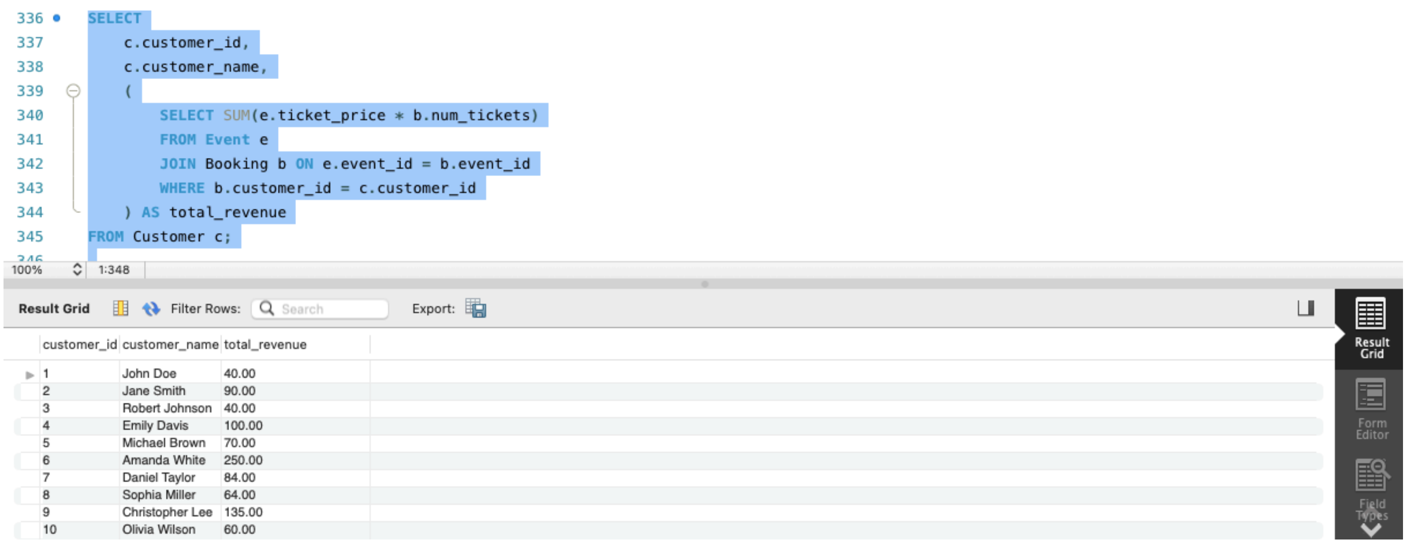
6. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.



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

```
event_id event_name ticket_pri...
3        Event 3    40.00
5        Event 5    35.00
6        Event 6    50.00
9        Event 9    45.00
```

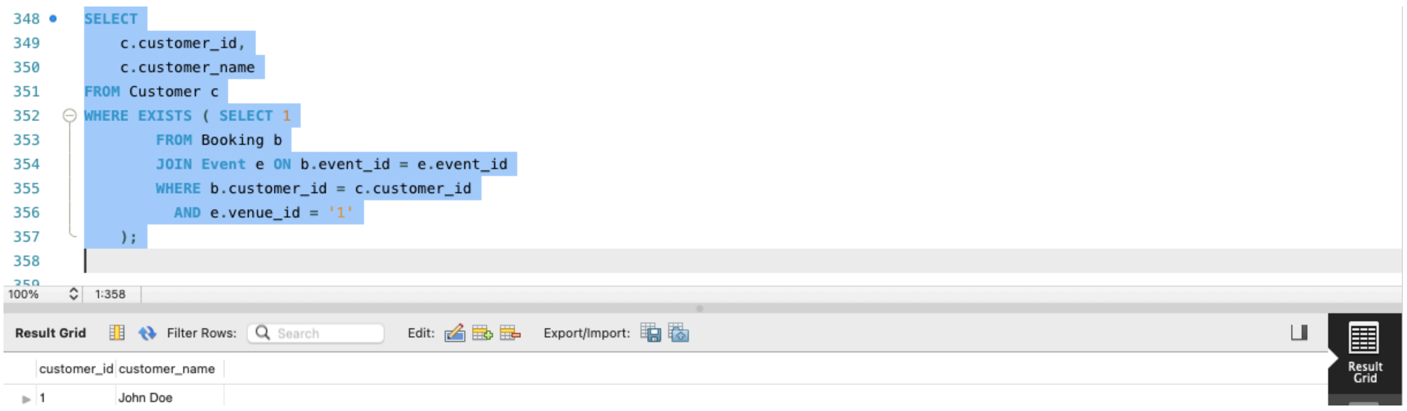
7. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.



```
SELECT
    c.customer_id,
    c.customer_name,
    (
        SELECT SUM(e.ticket_price * b.num_tickets)
        FROM Event e
        JOIN Booking b ON e.event_id = b.event_id
        WHERE b.customer_id = c.customer_id
    ) AS total_revenue
FROM Customer c;
```

```
customer_id customer_name   total_revenue
1           John Doe        40.00
2           Jane Smith      90.00
3           Robert Johnson  40.00
4           Emily Davis     100.00
5           Michael Brown   70.00
6           Amanda White    250.00
7           Daniel Taylor   84.00
8           Sophia Miller   64.00
9           Christopher Lee 135.00
10          Olivia Wilson   60.00
```

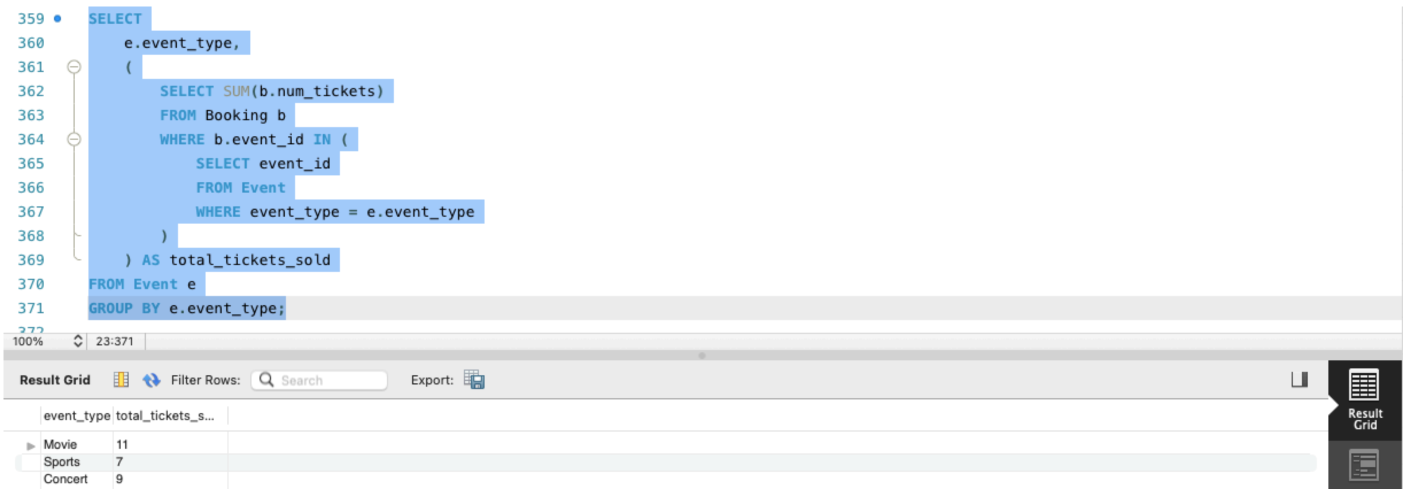
8. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.



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

```
customer_id customer_name
1           John Doe
```

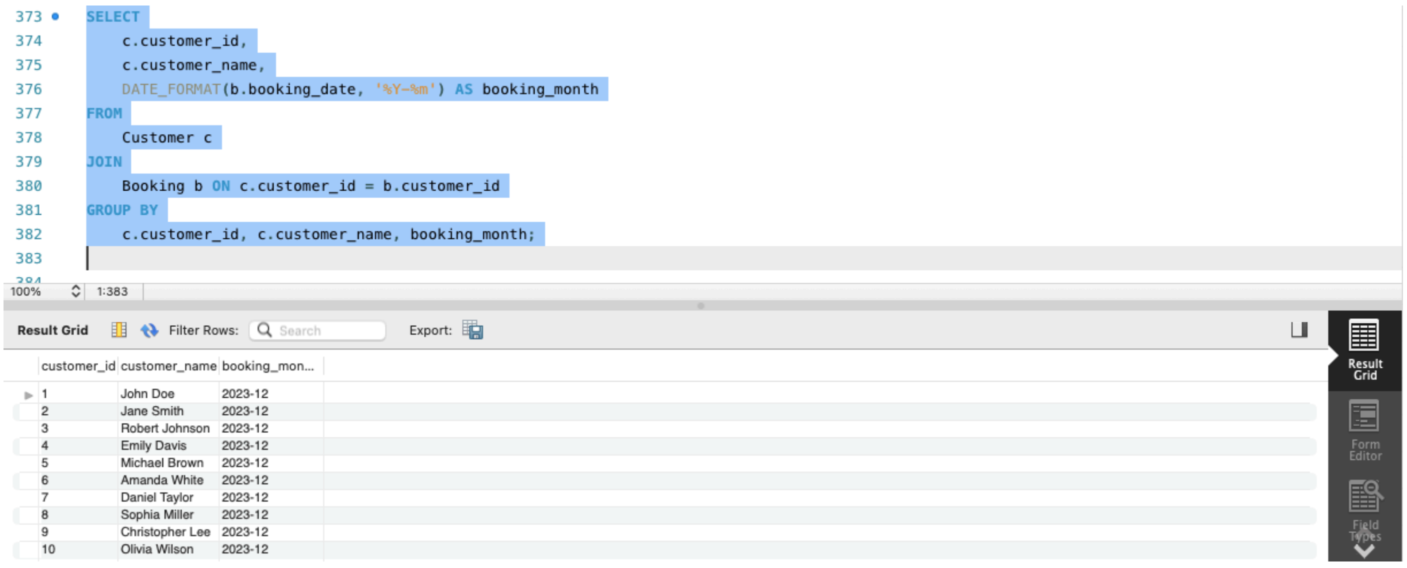
9. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY.



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

```
event_type  total_tickets_s...
Movie       11
Sports      7
Concert     9
```

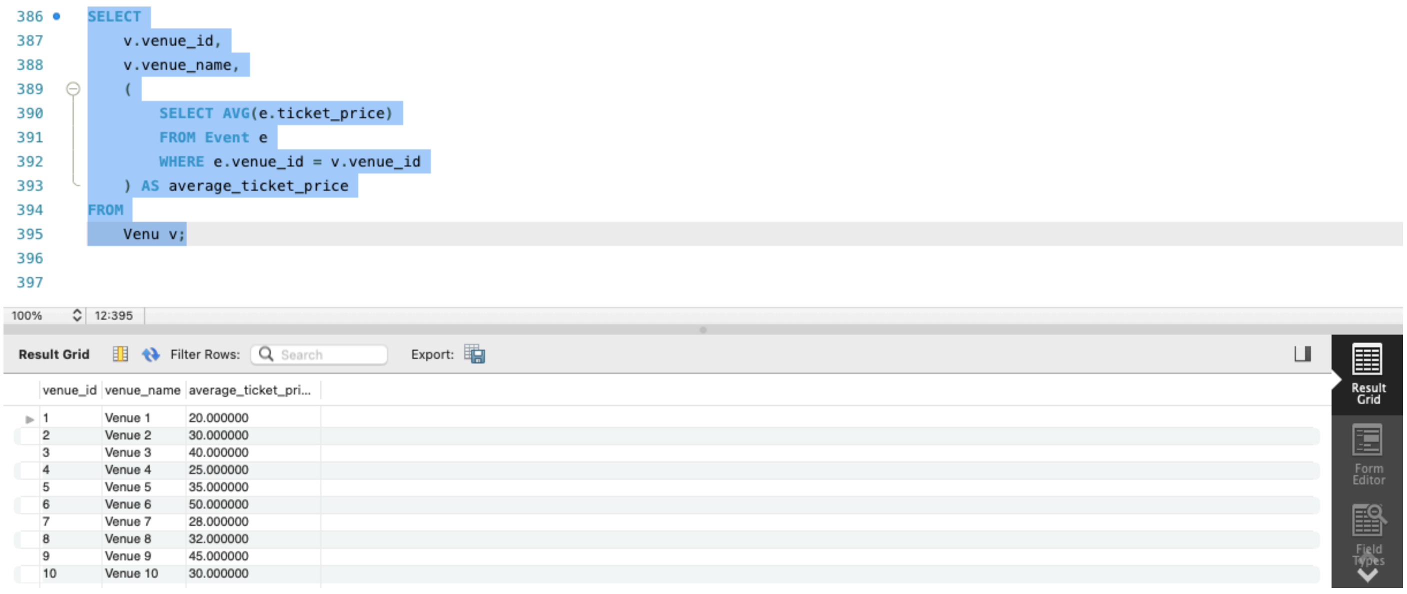
10. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT.



```sql
SELECT
    c.customer_id,
    c.customer_name,
    DATE_FORMAT(b.booking_date, '%Y-%m') AS booking_month
FROM
    Customer c
JOIN
    Booking b ON c.customer_id = b.customer_id
GROUP BY
    c.customer_id, c.customer_name, booking_month;
```

```
customer_id  customer_name    booking_mon...
1            John Doe         2023-12
2            Jane Smith       2023-12
3            Robert Johnson   2023-12
4            Emily Davis      2023-12
5            Michael Brown    2023-12
6            Amanda White     2023-12
7            Daniel Taylor    2023-12
8            Sophia Miller    2023-12
9            Christopher Lee  2023-12
10           Olivia Wilson    2023-12
```

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



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

```
venue_id  venue_name  average_ticket_pri...
1         Venue 1     20.000000
2         Venue 2     30.000000
3         Venue 3     40.000000
4         Venue 4     25.000000
5         Venue 5     35.000000
6         Venue 6     50.000000
7         Venue 7     28.000000
8         Venue 8     32.000000
9         Venue 9     45.000000
10        Venue 10    30.000000
```