TICKET_BOOKING_SYSTEM

Work by -

PAVITHRA B

Pavithrabala2203@gmail.com

Task 4

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

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

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

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

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

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

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

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

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

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

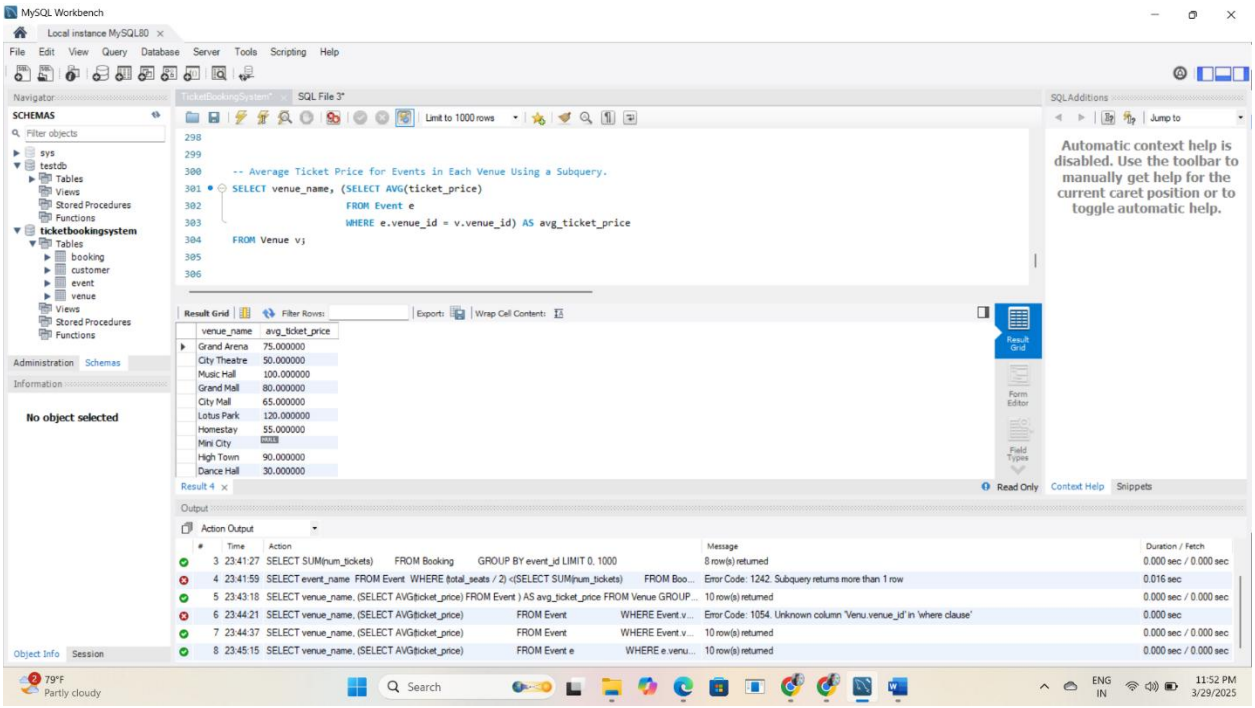
11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT.

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

Screenshots Of Task 4 :

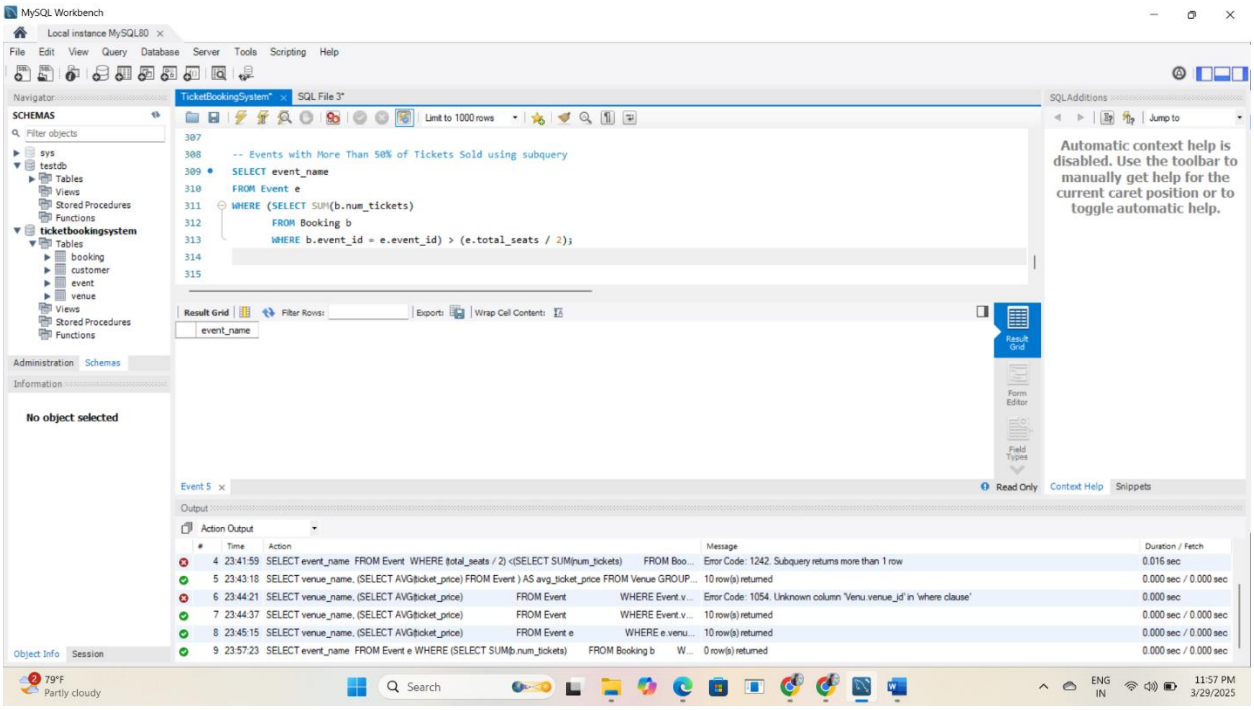
1.Average Ticket Price for Events in Each Venue Using a Subquery

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



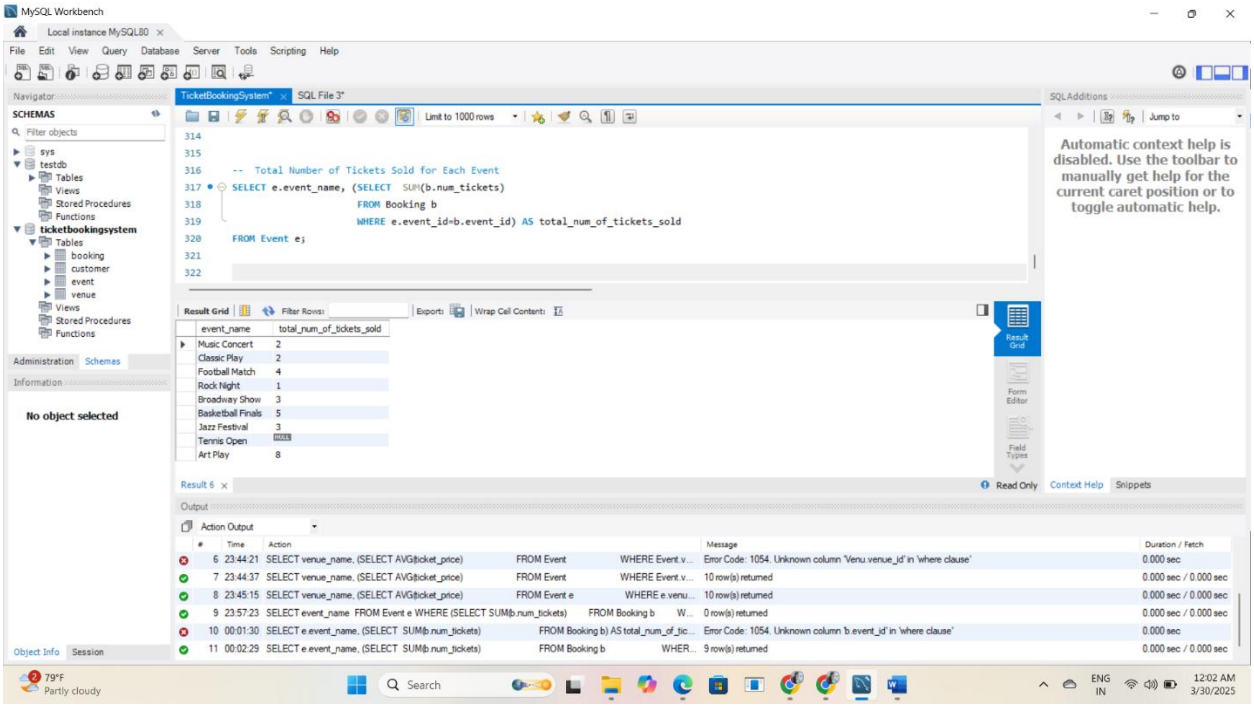
2.Events with More Than 50% of Tickets Sold using subquery

```
SELECT e.event_name
FROM Event e
WHERE (SELECT SUM(b.num_tickets)
FROM Booking b
 WHERE b.event_id = e.event_id) > (e.total_seats / 2);
```



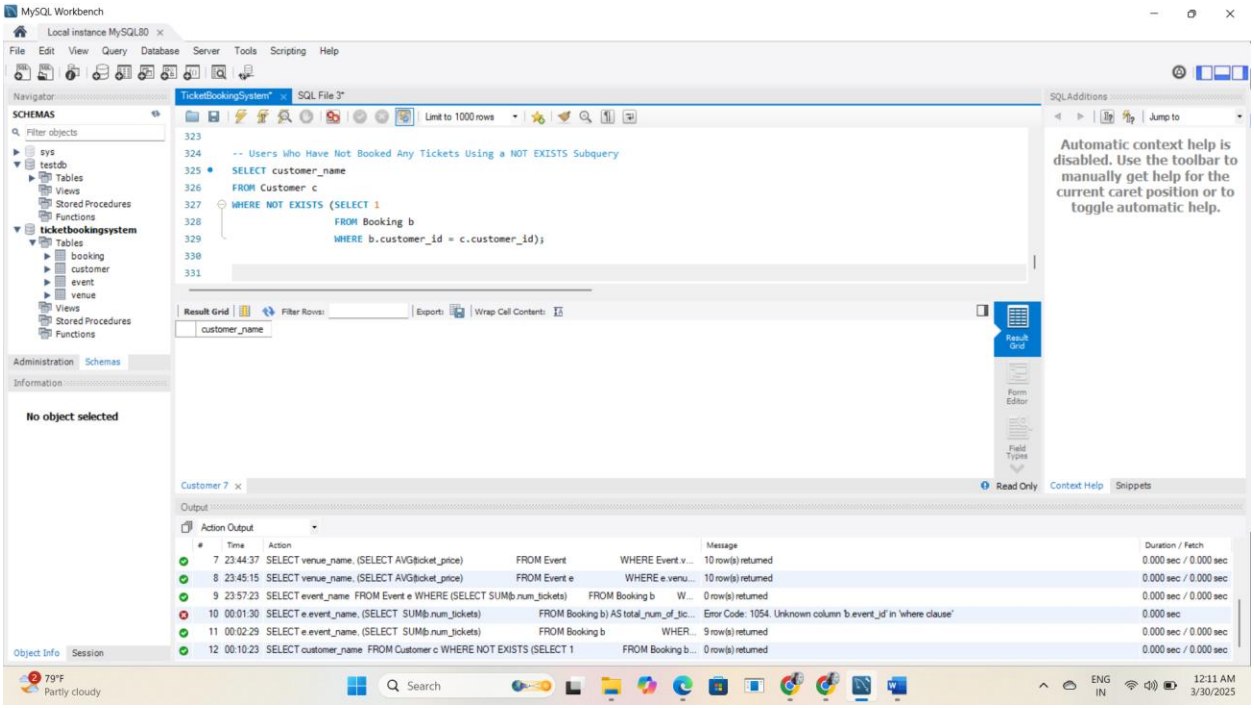
3. Total Number of Tickets Sold for Each Event

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



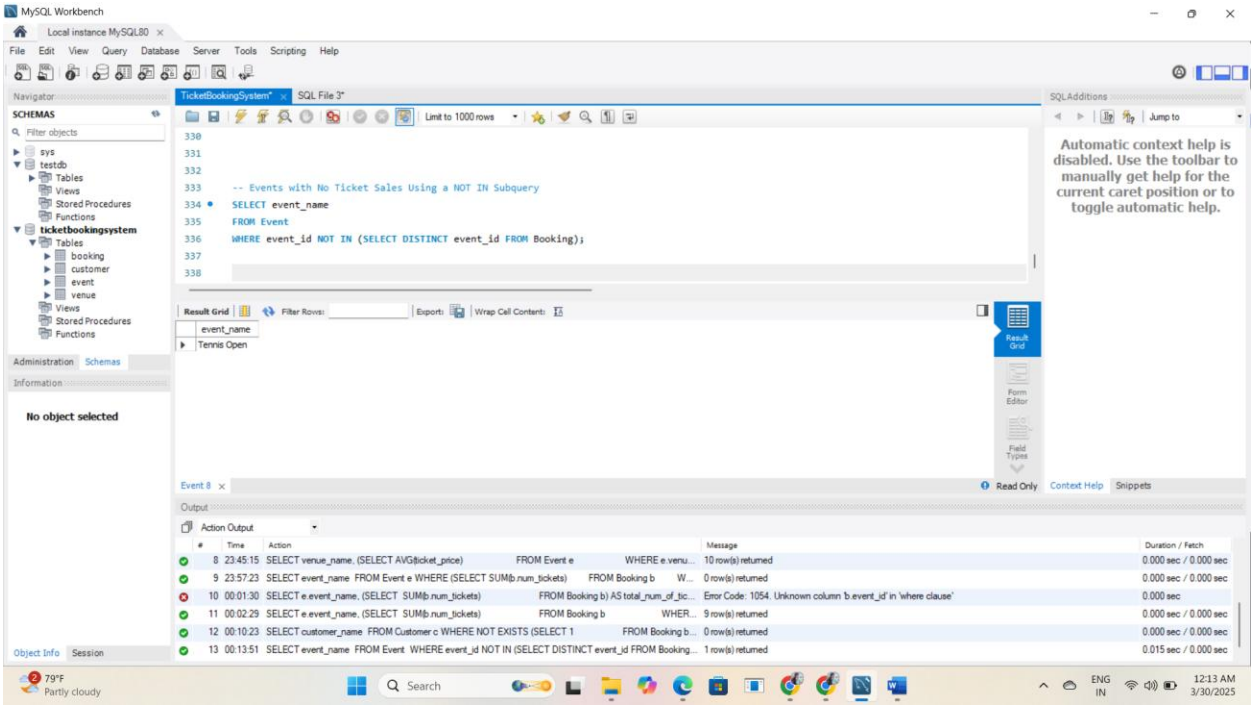
4. Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery

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



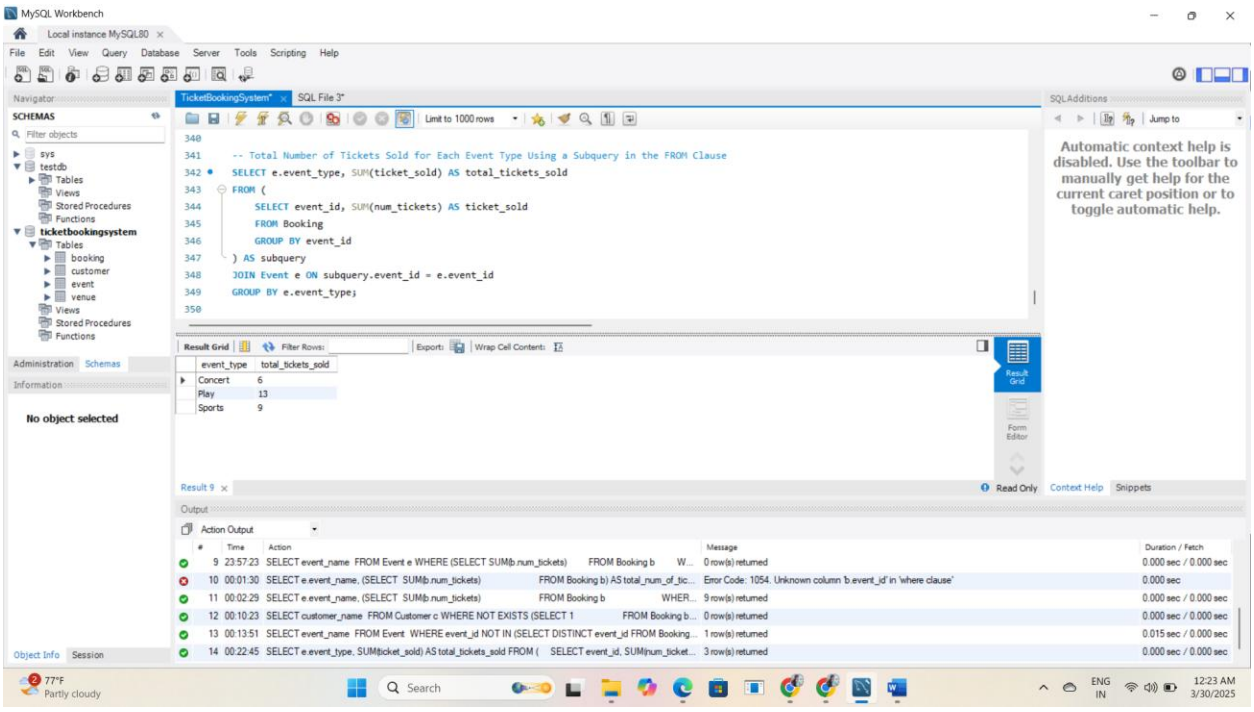
5. Events with No Ticket Sales Using a NOT IN Subquery

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



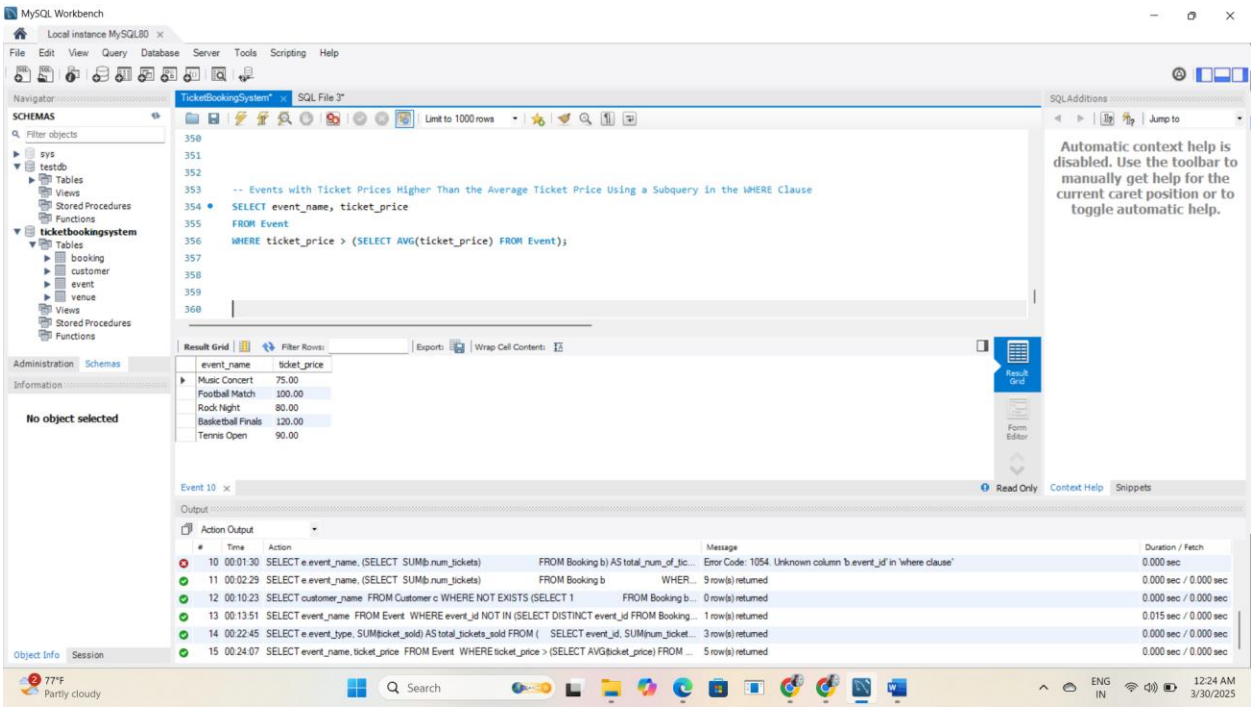
6. Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause

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



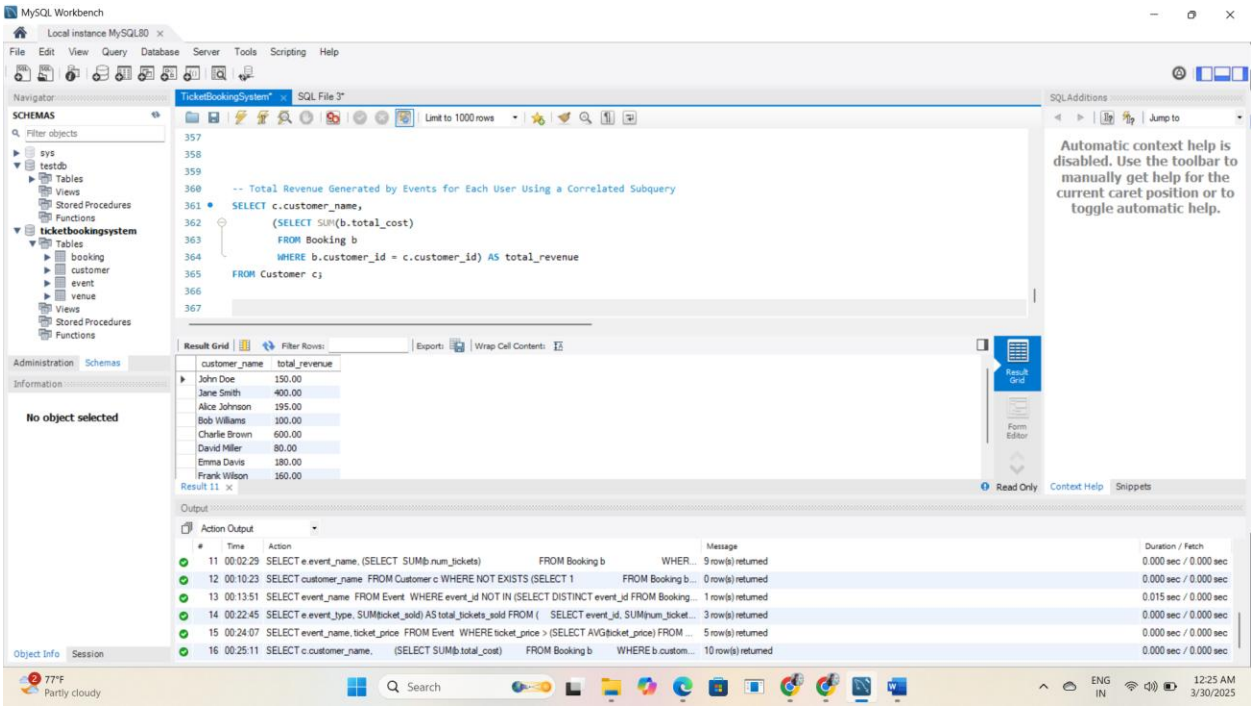
7. Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause

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



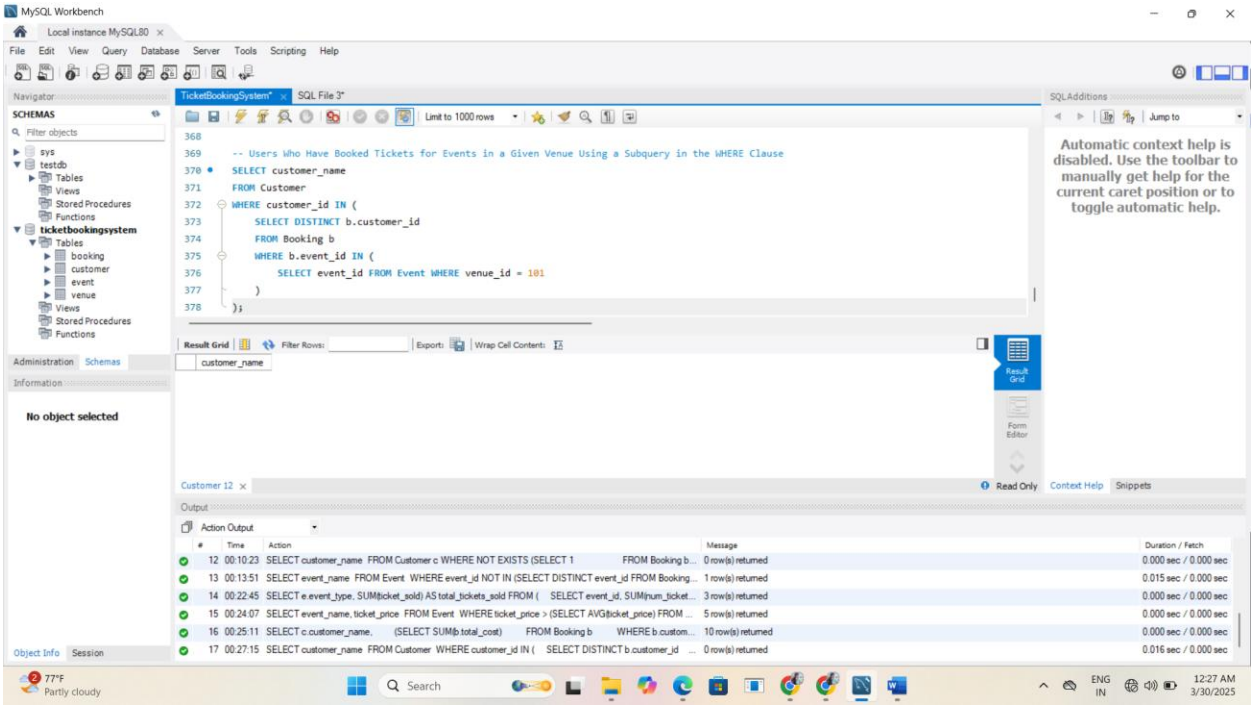
8. Total Revenue Generated by Events for Each User Using a Correlated Subquery

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



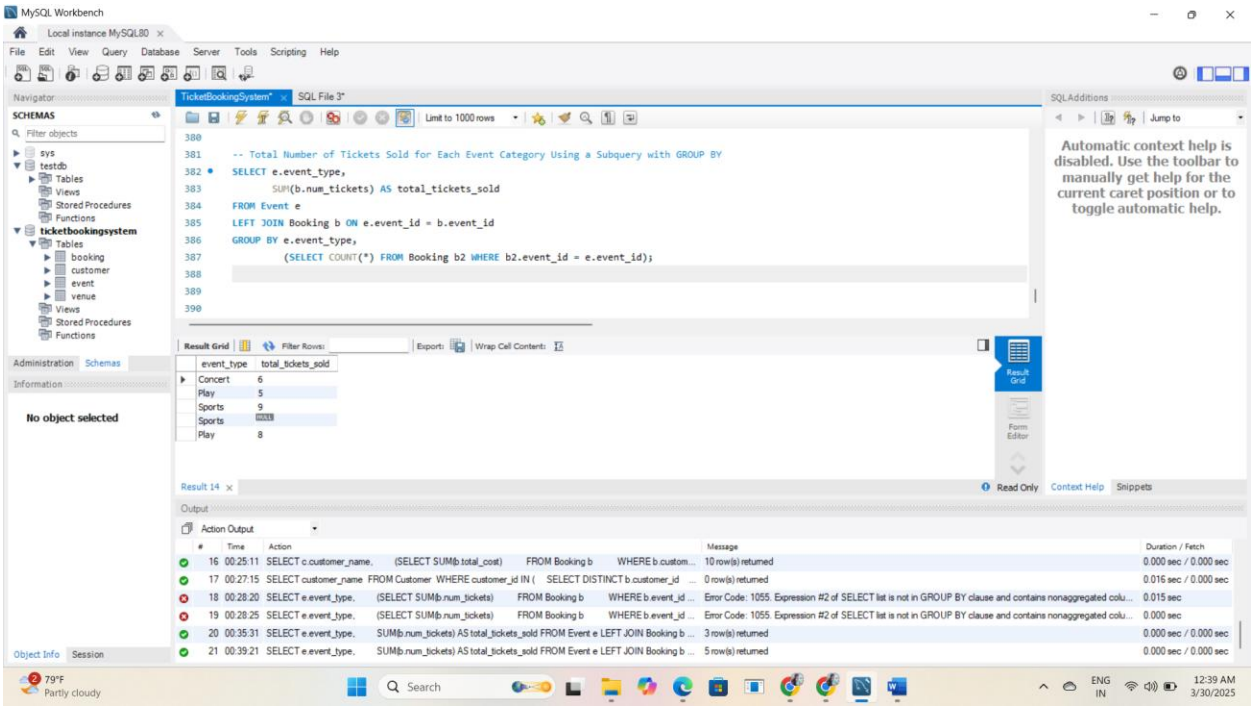
9. Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause

```
SELECT customer_name
FROM Customer
WHERE customer_id IN (
 SELECT DISTINCT b.customer_id
 FROM Booking b
WHERE b.event_id IN (
SELECT event_id FROM Event WHERE venue_id = 101
    )
);
```



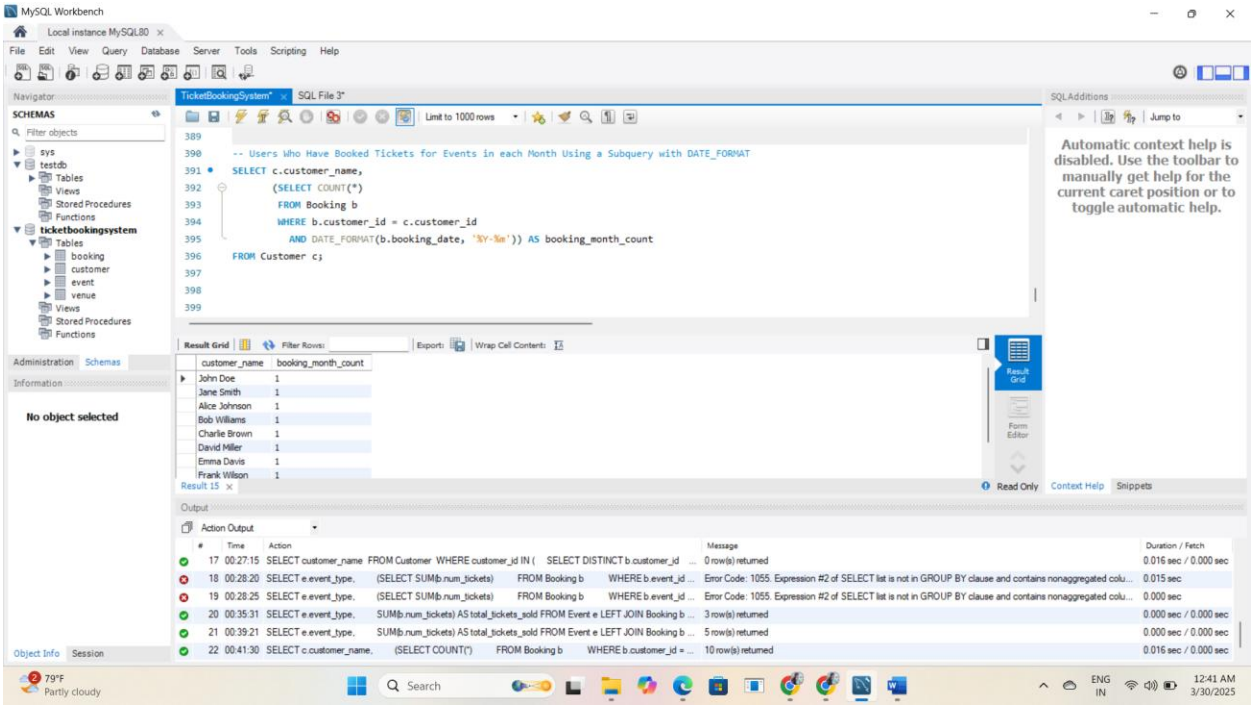
10. Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY

```
SELECT e.event_type,
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_type,
 (SELECT COUNT(*) FROM Booking b2 WHERE b2.event_id = e.event_id);
```



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

```
SELECT c.customer_name,
(SELECT COUNT(*)
FROM Booking b
WHERE b.customer_id = c.customer_id
AND DATE_FORMAT(b.booking_date, '%Y-%m')) AS booking_month_count
FROM Customer c;
```



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

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