

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;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
-- 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;
```

The Results window displays the following data:

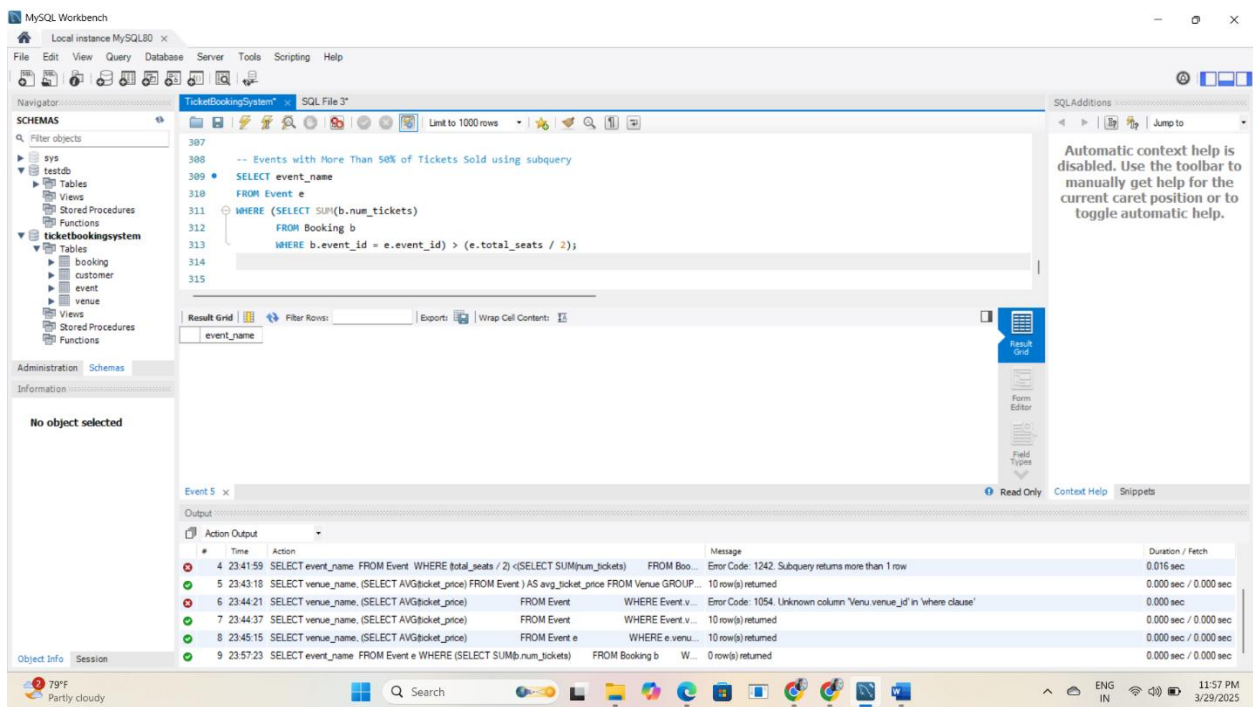
venue_name	avg_ticket_price
Grand Arena	75.000000
City Theatre	50.000000
Music Hall	100.000000
Grand Mall	80.000000
City Mall	65.000000
Lotus Park	120.000000
Homestay	55.000000
Mini City	60.000000
High Town	90.000000
Dance Hall	30.000000

The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
3	23:41:27	SELECT SUM(num_tickets) FROM Booking GROUP BY event_id LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
4	23:41:59	SELECT event_name FROM Event WHERE (total_seats / 2) < (SELECT SUM(num_tickets) FROM Booking GROUP BY event_id LIMIT 0, 1000)	Error Code: 1242. Subquery returns more than 1 row	0.016 sec
5	23:43:18	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event) AS avg_ticket_price FROM Venue GROUP BY venue_name	10 row(s) returned	0.000 sec / 0.000 sec
6	23:44:21	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event WHERE Event.venue_id = v.venue_id) AS avg_ticket_price FROM Venue v	Error Code: 1054. Unknown column 'Venue.venue_id' in 'where clause'	0.000 sec
7	23:44:37	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event WHERE Event.venue_id = v.venue_id) AS avg_ticket_price FROM Venue v	10 row(s) returned	0.000 sec / 0.000 sec
8	23:45:15	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event e WHERE e.venue_id = v.venue_id) AS avg_ticket_price FROM Venue v	10 row(s) returned	0.000 sec / 0.000 sec

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;
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
-- 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;
```

The Results Grid shows the following data:

event_name	total_num_of_tickets_sold
Music Concert	2
Classic Play	2
Football Match	4
Rock Night	1
Broadway Show	3
Basketball Finals	5
Jazz Festival	3
Tennis Open	3
Art Play	8

The Output tab shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
6	23:44:21	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event WHERE Event.v...	Error Code: 1054. Unknown column 'Venue venue_id' in 'where clause'	0.000 sec
7	23:44:37	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event WHERE Event.v...	10 row(s) returned	0.000 sec / 0.000 sec
8	23:45:15	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event e WHERE e venu...	10 row(s) returned	0.000 sec / 0.000 sec
9	23:57:23	SELECT event_name FROM Event e WHERE (SELECT SUM(b.num_tickets) FROM Booking b W...	0 row(s) returned	0.000 sec / 0.000 sec
10	00:01:30	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b) AS total_num_of_tic...	Error Code: 1054. Unknown column 'b.event_id' in 'where clause'	0.000 sec
11	00:02:29	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b WHERE...	9 row(s) returned	0.000 sec / 0.000 sec

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);
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
-- 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);
```

The Output tab shows the execution results:

#	Time	Action	Message	Duration / Fetch
7	23:44:37	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event e WHERE Event v...	10 row(s) returned	0.000 sec / 0.000 sec
8	23:45:15	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event e WHERE e venu...	10 row(s) returned	0.000 sec / 0.000 sec
9	23:57:23	SELECT event_name FROM Event e WHERE (SELECT SUM(b.num_tickets) FROM Booking b W...	0 row(s) returned	0.000 sec / 0.000 sec
10	00:01:30	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b) AS total_num_of_tic...	Error Code: 1054. Unknown column 'b.event_id' in 'where clause'	0.000 sec
11	00:02:29	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b WHERE...	9 row(s) returned	0.000 sec / 0.000 sec
12	00:10:23	SELECT customer_name FROM Customer c WHERE NOT EXISTS (SELECT 1 FROM Booking b...	0 row(s) returned	0.000 sec / 0.000 sec

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);

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
-- 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);
```

The Results Grid shows the following data:

event_name
Tennis Open

The Output tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
8	23:45:15	SELECT venue_name, (SELECT AVG(ticket_price) FROM Event e WHERE e.venue_id = b.venue_id) FROM Booking b	10 row(s) returned	0.000 sec / 0.000 sec
9	23:57:23	SELECT event_name FROM Event e WHERE (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id) = 0	0 row(s) returned	0.000 sec / 0.000 sec
10	00:01:30	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b AS total_num_of_tickets WHERE b.event_id = e.event_id) FROM Event e	Error Code: 1054. Unknown column 'b.event_id' in 'where clause'	0.000 sec
11	00:02:29	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id) FROM Event e	9 row(s) returned	0.000 sec / 0.000 sec
12	00:10:23	SELECT customer_name FROM Customer c WHERE NOT EXISTS (SELECT 1 FROM Booking b WHERE b.customer_id = c.customer_id)	0 row(s) returned	0.000 sec / 0.000 sec
13	00:13:51	SELECT event_name FROM Event WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking)	1 row(s) returned	0.015 sec / 0.000 sec

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;
```

The screenshot displays the MySQL Workbench interface. The SQL Editor window contains the following query:

```
-- Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause
342 SELECT e.event_type, SUM(ticket_sold) AS total_tickets_sold
343 FROM (
344     SELECT event_id, SUM(num_tickets) AS ticket_sold
345     FROM Booking
346     GROUP BY event_id
347 ) AS subquery
348 JOIN Event e ON subquery.event_id = e.event_id
349 GROUP BY e.event_type;
```

The Results Grid shows the output of the query:

event_type	total_tickets_sold
Concert	6
Play	13
Sports	9

The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
9	23:57:23	SELECT event_name FROM Event e WHERE (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id)	0 row(s) returned	0.000 sec / 0.000 sec
10	00:01:30	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b) AS total_num_of_tickets FROM Event e	Error Code: 1054. Unknown column 'b.event_id' in 'where clause'	0.000 sec
11	00:02:29	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b) AS total_num_of_tickets FROM Event e	9 row(s) returned	0.000 sec / 0.000 sec
12	00:10:23	SELECT customer_name FROM Customer c WHERE NOT EXISTS (SELECT 1 FROM Booking b WHERE b.customer_id = c.customer_id)	0 row(s) returned	0.000 sec / 0.000 sec
13	00:13:51	SELECT event_name FROM Event WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking)	1 row(s) returned	0.015 sec / 0.000 sec
14	00:22:45	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	3 row(s) returned	0.000 sec / 0.000 sec

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);

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
-- 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);
```

The Results window displays the following data:

event_name	ticket_price
Music Concert	75.00
Football Match	100.00
Rock Night	80.00
Basketball Finals	120.00
Tennis Open	90.00

The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
10	00:01:30	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b) AS total_num_of_tickets	Error Code: 1054. Unknown column 'b.event_id' in 'where clause'	0.000 sec
11	00:02:29	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b) WHERE...	9 row(s) returned	0.000 sec / 0.000 sec
12	00:10:23	SELECT customer_name FROM Customer c WHERE NOT EXISTS (SELECT 1 FROM Booking b...	0 row(s) returned	0.000 sec / 0.000 sec
13	00:13:51	SELECT event_name FROM Event WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking...	1 row(s) returned	0.015 sec / 0.000 sec
14	00:22:45	SELECT e.event_type, SUM(b.ticket_sold) AS total_tickets_sold FROM (SELECT event_id, SUM(num_ticket...	3 row(s) returned	0.000 sec / 0.000 sec
15	00:24:07	SELECT event_name, ticket_price FROM Event WHERE ticket_price > (SELECT AVG(ticket_price) FROM ...	5 row(s) returned	0.000 sec / 0.000 sec

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;
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
-- 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;
```

The Results Grid shows the output of the query:

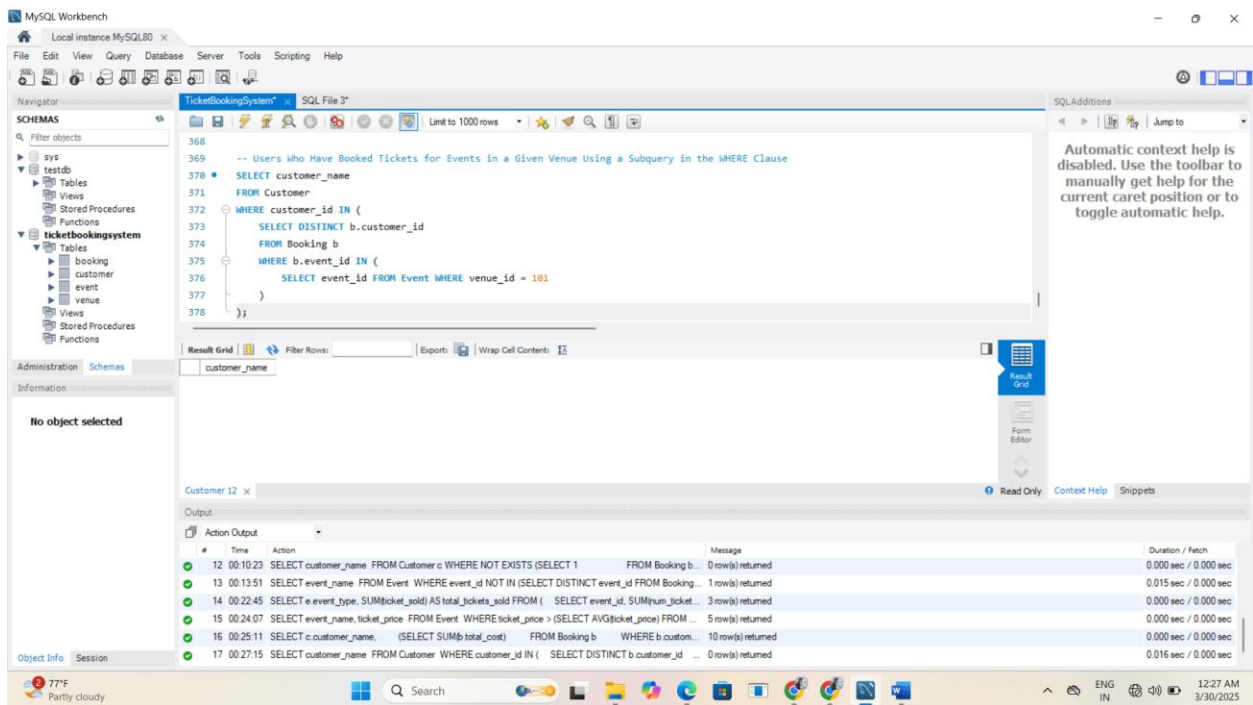
customer_name	total_revenue
John Doe	150.00
Jane Smith	400.00
Alice Johnson	295.00
Bob Williams	300.00
Charlie Brown	600.00
David Miller	80.00
Emma Davis	180.00
Frank Wilson	160.00

The Output tab shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
11	00:02:29	SELECT e.event_name, (SELECT SUM(b.num_tickets) FROM Booking b WHERE e.event_id = b.event_id) AS total_tickets_sold FROM Event e	9 row(s) returned	0.000 sec / 0.000 sec
12	00:10:23	SELECT customer_name FROM Customer c WHERE NOT EXISTS (SELECT 1 FROM Booking b WHERE b.customer_id = c.customer_id)	0 row(s) returned	0.000 sec / 0.000 sec
13	00:13:51	SELECT event_name FROM Event WHERE event_id NOT IN (SELECT DISTINCT event_id FROM Booking)	1 row(s) returned	0.015 sec / 0.000 sec
14	00:22:45	SELECT e.event_type, SUM(b.ticket_sold) AS total_tickets_sold FROM (SELECT event_id, SUM(num_ticket) FROM Booking b GROUP BY event_id) b JOIN Event e ON b.event_id = e.event_id	3 row(s) returned	0.000 sec / 0.000 sec
15	00:24:07	SELECT event_name, ticket_price FROM Event WHERE ticket_price > (SELECT AVG(ticket_price) FROM Event)	5 row(s) returned	0.000 sec / 0.000 sec
16	00:25:11	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	10 row(s) returned	0.000 sec / 0.000 sec

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);
```

The screenshot displays the MySQL Workbench interface. The SQL Editor window contains the following query:

```
-- 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);
```

The Results window shows the output of the query:

event_type	total_tickets_sold
Concert	6
Play	5
Sports	9
Play	8

The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
16	00:25:11	SELECT c.customer_name, (SELECT SUM(b.total_cost) FROM Booking b WHERE b.customer_id = c.customer_id) AS total_cost FROM Customer c	10 row(s) returned	0.000 sec / 0.000 sec
17	00:27:15	SELECT customer_name FROM Customer WHERE customer_id IN (SELECT DISTINCT b.customer_id FROM Booking b)	0 row(s) returned	0.016 sec / 0.000 sec
18	00:28:20	SELECT e.event_type, (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id) AS total_tickets_sold FROM Event e	Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'event_type' which is not functionally dependent on this table's partitioning function.	0.015 sec
19	00:28:25	SELECT e.event_type, (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id) AS total_tickets_sold FROM Event e	Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'event_type' which is not functionally dependent on this table's partitioning function.	0.000 sec
20	00:35:31	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	3 row(s) returned	0.000 sec / 0.000 sec
21	00:39:21	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	5 row(s) returned	0.000 sec / 0.000 sec

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;
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
-- 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;
```

The Results Grid shows the following data:

customer_name	booking_month_count
John Doe	1
Jane Smith	1
Alice Johnson	1
Bob Williams	1
Charlie Brown	1
David Miller	1
Emma Davis	1
Frank Wilson	1

The Output tab shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
17	00:27:15	SELECT customer_name FROM Customer WHERE customer_id IN (SELECT DISTINCT b.customer_id ...	0 row(s) returned	0.016 sec / 0.000 sec
18	00:28:20	SELECT e.event_type, (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id ...	Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated colu...	0.015 sec
19	00:28:25	SELECT e.event_type, (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id ...	Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated colu...	0.000 sec
20	00:35:31	SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold FROM Event e LEFT JOIN Booking b ...	3 row(s) returned	0.000 sec / 0.000 sec
21	00:39:21	SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold FROM Event e LEFT JOIN Booking b ...	5 row(s) returned	0.000 sec / 0.000 sec
22	00:41:30	SELECT c.customer_name, (SELECT COUNT(*) FROM Booking b WHERE b.customer_id = ...	10 row(s) returned	0.016 sec / 0.000 sec

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;
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
-- 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;
```

The Results window displays the output of the query:

venue_name	avg_ticket_price
Grand Arena	75.000000
City Theatre	50.000000
Music Hall	100.000000
Grand Mall	80.000000
City Mall	65.000000
Lotus Park	120.000000
Homestay	55.000000
Min City	0.000000

The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
18	00:28:20	SELECT e.event_type, (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = ...	Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated colu...	0.015 sec
19	00:28:25	SELECT e.event_type, (SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = ...	Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated colu...	0.000 sec
20	00:35:31	SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold FROM Event e LEFT JOIN Booking b ...	3 row(s) returned	0.000 sec / 0.000 sec
21	00:39:21	SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold FROM Event e LEFT JOIN Booking b ...	5 row(s) returned	0.000 sec / 0.000 sec
22	00:41:30	SELECT e.customer_name, (SELECT COUNT(*) FROM Booking b WHERE b.customer_id = ...	10 row(s) returned	0.016 sec / 0.000 sec
23	00:42:23	SELECT v.venue_name, (SELECT AVG(e.ticket_price) FROM Event e WHERE e.venue_id = ...	10 row(s) returned	0.000 sec / 0.000 sec