**Task 4: Subquery and its types**

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

Query:

SELECT venue\_id,

(SELECT AVG(ticket\_price) FROM Event e WHERE e.venue\_id = v.venue\_id) AS avg\_ticket\_price

FROM Venue v;

Output:

+----------+----------------- +

| venue\_id | avg\_ticket\_price |

+----------+----------------- +

| 1 | 500.00 |

| 2 | 1500.00 |

| 3 | 300.00 |

| 4 | 700.00 |

+----------+----------------- +

2. Find Events with More Than 50% of Tickets Sold Using Subquery

Query:

SELECT event\_name, total\_seats, available\_seats

FROM Event

WHERE available\_seats < (total\_seats / 2);

Output:

+-----------------+------------- +---------------- +

| event\_name | total\_seats | available\_seats |

+-----------------+------------- +---------------- +

| Movie Night | 200 | 150 |

| Bhopal Carnival | 1000 | 800 |

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

Query:

SELECT event\_name, total\_seats - available\_seats AS tickets\_sold

FROM Event;

Output:

+-----------------+--------------+

| event\_name | tickets\_sold |

+-----------------+--------------+

| Movie Night | 50 |

| Cricket Match | 1000 |

| Music Evening | 20 |

| Bhopal Carnival | 200 |

+-----------------+--------------+

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

Query:

SELECT customer\_name

FROM Customer c

WHERE NOT EXISTS (SELECT 1 FROM Booking b WHERE b.customer\_id = c.customer\_id);

Output:

No records found. All users have booked tickets.

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

Query:

SELECT event\_name

FROM Event

WHERE event\_id NOT IN (SELECT event\_id FROM Booking);

Output:

No events found. All events have sales.

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

Query:

SELECT event\_type, SUM(tickets\_sold) AS total\_tickets\_sold

FROM (SELECT event\_type, total\_seats - available\_seats AS tickets\_sold FROM Event) AS EventSummary

GROUP BY event\_type;

Output:

+-----------+-------------------+

| event\_type| total\_tickets\_sold |

+-----------+-------------------+

| Movie | 50 |

| Sports | 1000 |

| Concert | 220 |

+-----------+-------------------+

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

Query:

SELECT event\_name, ticket\_price

FROM Event

WHERE ticket\_price > (SELECT AVG(ticket\_price) FROM Event);

Output:

+----------------+--------------+

| event\_name | ticket\_price |

+----------------+--------------+

| Cricket Match | 1500.00 |

| Bhopal Carnival| 700.00 |

+----------------+--------------+

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

Query:

SELECT customer\_name,

(SELECT SUM(total\_cost)

FROM Booking b

WHERE b.customer\_id = c.customer\_id) AS total\_revenue

FROM Customer c;

Output:

+------------------+---------------+

| customer\_name | total\_revenue |

+------------------+---------------+

| Aditi Mishra | 1000.00 |

| Rohan Verma | 7500.00 |

| Neha Pandey | 300.00 |

| Vikrant Shukla | 2100.00 |

+------------------+---------------+

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

Query:

SELECT customer\_name

FROM Customer c

WHERE EXISTS (SELECT 1

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

WHERE e.venue\_id = 1 AND b.customer\_id = c.customer\_id);

Output:

+----------------+

| customer\_name |

+----------------+

| Aditi Mishra |

+----------------+

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

Query:

SELECT event\_type, SUM(tickets\_sold) AS total\_tickets\_sold

FROM (SELECT event\_type, total\_seats - available\_seats AS tickets\_sold FROM Event) AS EventSummary

GROUP BY event\_type;

Output:

+-----------+-------------------+

| event\_type| total\_tickets\_sold |

+-----------+-------------------+

| Movie | 50 |

| Sports | 1000 |

| Concert | 220 |

+-----------+-------------------+

11. Find Users Who Have Booked Tickets for Events in Each Month Using a Subquery with DATE\_FORMAT:

Query//

SELECT c.customer\_name,DATE\_FORMAT(b.booking\_date, '%Y-%m') AS booking\_month FROM Customer c JOIN Booking b ON c.booking\_id = b.booking\_id

WHERE EXISTS (

SELECT 1

FROM Booking b\_sub

WHERE b\_sub.customer\_id = b.customer\_id

AND DATE\_FORMAT(b\_sub.booking\_date, '%Y-%m') = DATE\_FORMAT(b.booking\_date, '%Y-%m')

)

GROUP BY c.customer\_name, booking\_month

ORDER BY booking\_month;

Output:

+-----------------+---------------+

| customer\_name | booking\_month |

+-----------------+---------------+

| Aditi Mishra | 2024-10 |

| Rohan Verma | 2024-10 |

| Neha Pandey | 2024-10 |

| Vikrant Shukla | 2024-10 |

+-----------------+---------------+

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

//Query//

SELECT v.venue\_name,AVG(e.ticket\_price) AS average\_ticket\_price

FROM Venue v JOIN Event e ON v.venue\_id = e.venue\_id GROUP BY v.venue\_name;

Output:

+-------------------+----------------------+

| venue\_name | average\_ticket\_price |

+-------------------+----------------------+

| Shree Ram Theatre | 500.000000 |

| Bhopal Haat | 1500.000000 |

| Vidisha Heritage | 300.000000 |

| Kamal Palace | 700.000000 |

|  |  |
| --- | --- |
|  |  |

END OF ALL TASKS