**Tasks 4: Subquery and its types**

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

select v.venue\_name ,(select avg(e.ticket\_price) from Event\_Table e WHERE e.venue\_id = v.venue\_id)as averageticketprice from Venue\_Tables v;

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

| venue\_name | averageticketprice |

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

| Theatre Arena | 1000.000000 |

| Silver Screen | 2000.000000 |

| Movie magic | 4000.000000 |

| Music Venue | 6000.000000 |

| sound stage | 1500.000000 |

| Classic Fusion | 3000.000000 |

| Game Stadium | 2500.000000 |

| Sports Venue | 3500.000000 |

| playground Arena | 500.000000 |

| champion field | 4500.000000 |

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

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

select e.event\_id , e.event\_name from Event\_Table e WHERE (select sum(b.num\_tickets) from Booking\_Table b WHERE e.event\_id = b.event\_id ) > (e.total\_seats/2);

Empty set

1. **Calculate the Total Number of Tickets Sold for Each Event.**

select e.event\_name , e.event\_id , (select sum(b.num\_tickets) from Booking\_Table b WHERE e.event\_id=b.event\_id) as totaltickets from Event\_Table e;

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

| event\_name | event\_id | totaltickets |

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

| Avengers The End Game | 201 | 1 |

| Interstellar | 202 | 2 |

| Fault in our stars | 203 | 3 |

| Folk | 204 | 1 |

| Melody Evening | 205 | 4 |

| Rythm Party | 206 | 2 |

| Basket Ball | 207 | 1 |

| Fun Run Challenge | 208 | 5 |

| Cricket | 209 | 1 |

| Football | 210 | 2 |

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

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

select \* from Customer\_Table c WHERE NOT EXISTS (select 1 from Booking\_Table b WHERE c.customer\_id = b.customer\_id );

Empty set

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

select e.event\_id , e.event\_name from Event\_Table e WHERE event\_id NOT IN (select e.event\_id from Booking\_Table b);

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

select e.event\_type , sum(b.num\_tickets) as totaltickets FROM Event\_Table e JOIN (select event\_id , num\_tickets from Booking\_Table ) AS b ON e.event\_id = b.event\_id GROUP BY e.event\_type;

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

| event\_type | totaltickets |

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

| Movie | 6 |

| Concert | 7 |

| Sports | 9 |

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

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

select event\_id , event\_name from Event\_Table where ticket\_price > (select avg(ticket\_price) from Event\_Table );

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

| event\_id | event\_name |

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

| 203 | Fault in our stars |

| 204 | Folk |

| 206 | Rythm Party |

| 208 | Fun Run Challenge |

| 210 | Football |

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

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

select c.customer\_id , c.customer\_name from Customer\_Table c WHERE (select sum(b.total\_cost) from Booking\_Table b WHERE b.customer\_id = c.customer\_id) ;

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

| customer\_id | customer\_name |

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

| 301 | priya |

| 302 | preethi |

| 303 | ram |

| 304 | raghul |

| 305 | raj |

| 306 | ramya |

| 307 | anjali |

| 308 | harish |

| 309 | arjun |

| 310 | ramesh |

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

**9. 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\_Table c WHERE customer\_id IN (select b.customer\_id from Booking\_Table b JOIN Event\_Table e ON e.event\_id = b.event\_id WHERE e.venue\_id = 101);

Empty set (0.01 sec)

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

select e.event\_type , (select sum(b.num\_tickets) from Booking\_Table b WHERE e.event\_id = b.event\_id) AS totaltickets from Event\_Table e GROUP BY e.event\_type;

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

select c.customer\_id , c.customer\_name from Customer\_Table c where exists (select 1 from Booking\_Table b JOIN Event\_Table e ON e.event\_id = b.event\_id WHERE c.customer\_id = b.customer\_id AND DATE\_FORMAT(e.event\_date , '%y-%m') = '2025-03' );

Empty set

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

select v.venue\_name , (select avg( CASE WHEN e.ticket\_price IS NULL then 0 ELSE e.ticket\_price END )from Event\_Table e where e.venue\_id = v.venue\_id ) as avgticketprice from Venue\_Tables v;

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

| venue\_name | avgticketprice |

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

| Theatre Arena | 1000.000000 |

| Silver Screen | 2000.000000 |

| Movie magic | 4000.000000 |

| Music Venue | 6000.000000 |

| sound stage | 1500.000000 |

| Classic Fusion | 3000.000000 |

| Game Stadium | 2500.000000 |

| Sports Venue | 3500.000000 |

| playground Arena | 500.000000 |

| champion field | 4500.000000 |