**JAVA FOUNDATION TRAINING**

**BATCH 8**

**Name:** Rajalakshmi Ganesh             **Case Study:** TicketBookingSystem

**Tasks 4: Subquery and its types**

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

**🡪** SELECT venue\_id,venue\_name,

(SELECT AVG(ticket\_price) FROM event e

WHERE e.venue\_id=v.venue\_id) AS avg\_ticket\_price

FROM venue v;



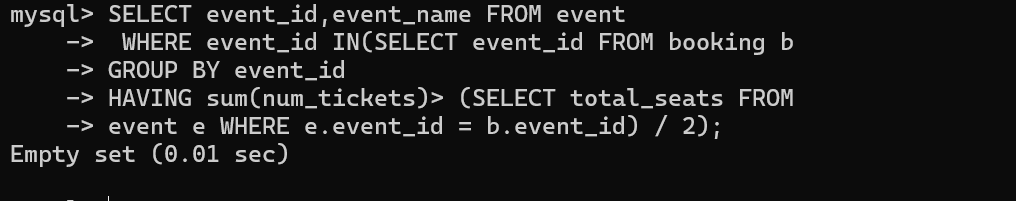
**2. Find Events with More Than 50% of Tickets Sold using subquery.**

**🡪** SELECT event\_id,event\_name FROM event

WHERE event\_id IN(SELECT event\_id FROM booking b

GROUP BY event\_id  
HAVING sum(num\_tickets)> (SELECT total\_seats FROM

event e WHERE e.event\_id = b.event\_id) / 2);

****

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

**🡪** SELECT event\_id,event\_name, (SELECT SUM(num\_tickets) from booking b where b.event\_id=e.event\_id ) AS total\_tickets\_sold FROM event e;

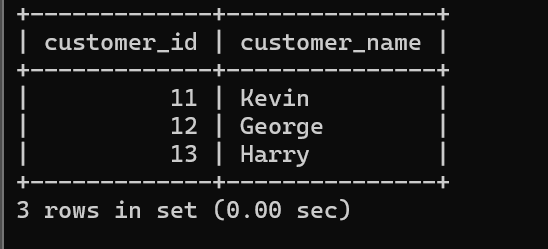
****

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

**🡪** 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);



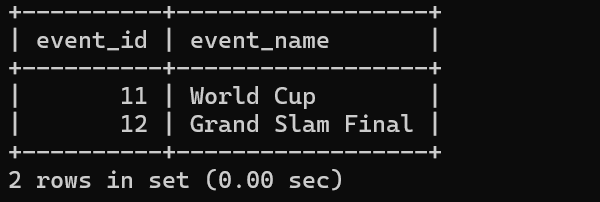
**5. List Events with No Ticket Sales Using a NOT IN Subquery.**

**🡪** SELECT event\_id,event\_name

FROM event

WHERE event\_id NOT IN (

SELECT DISTINCT event\_id from booking);

****

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

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

FROM (

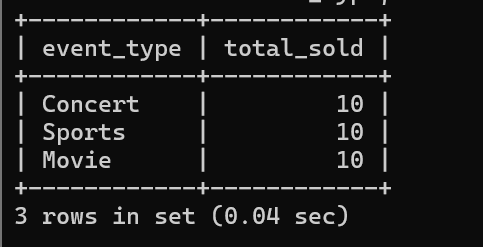
SELECT e.event\_type, SUM(b.num\_tickets) AS total\_tickets\_sold

FROM booking b

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

GROUP BY e.event\_type ) AS subquery

GROUP BY event\_type;

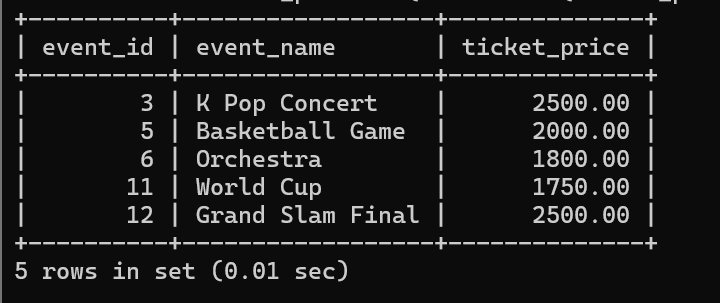
****

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

**🡪** SELECT event\_id, event\_name, ticket\_price

FROM event

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



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

**🡪** SELECT c.customer\_id, c.customer\_name, (

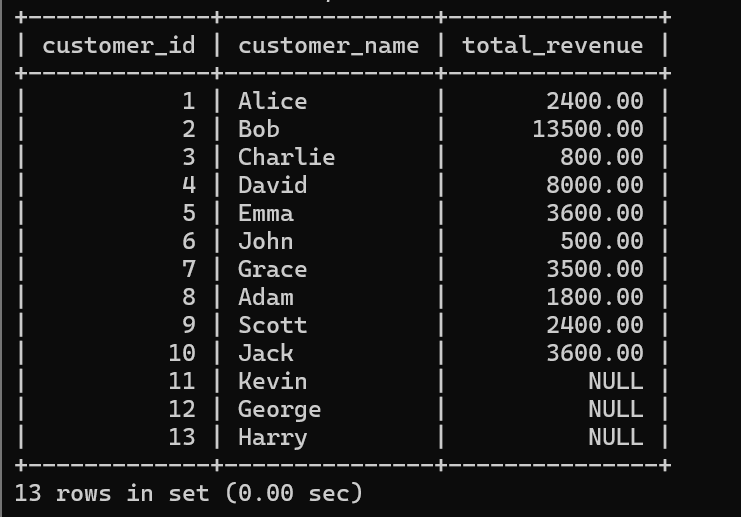
SELECT SUM(b.num\_tickets \* e.ticket\_price)

FROM booking b

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

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

FROM customer c;



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

**🡪** SELECT customer\_id, customer\_name

FROM customer

WHERE customer\_id IN (

SELECT DISTINCT b.customer\_id

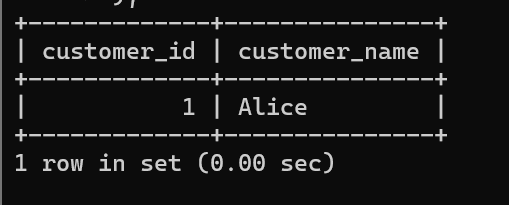
FROM booking b

WHERE b.event\_id IN (

SELECT e.event\_id

FROM event e

WHERE e.venue\_id = 1 ) );



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

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

FROM (

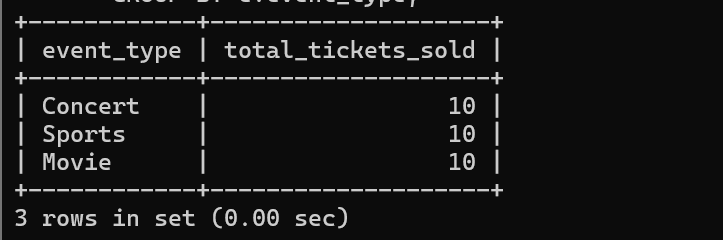
SELECT e.event\_type, SUM(b.num\_tickets) AS total\_tickets\_sold

FROM booking b

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

GROUP BY e.event\_type ) AS subquery

GROUP BY event\_type;

****

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

**🡪** SELECT booking\_month, customer\_id, customer\_name, total\_bookings

FROM (

SELECT DATE\_FORMAT(b.booking\_date, '%Y-%m') AS booking\_month, c.customer\_id, c.customer\_name,

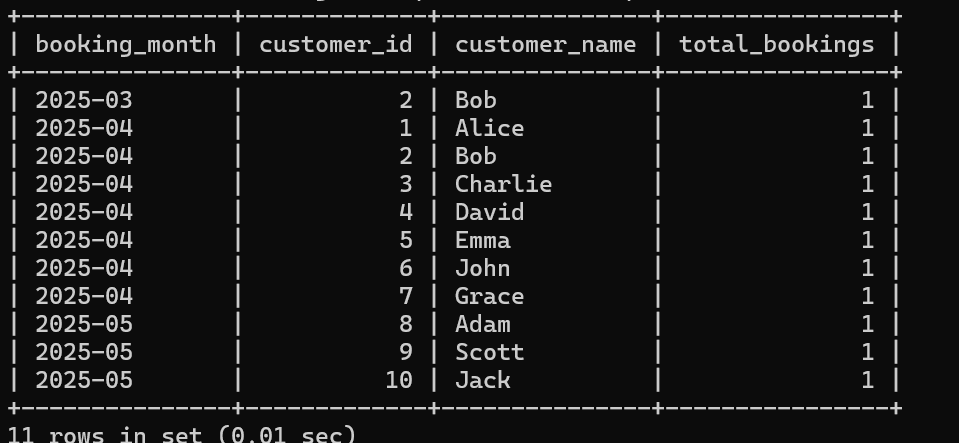
COUNT(b.booking\_id) AS total\_bookings

FROM customer c

JOIN booking b ON c.customer\_id = b.customer\_id

GROUP BY booking\_month, c.customer\_id, c.customer\_name) AS subquery

ORDER BY booking\_month, customer\_id;



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

**🡪** SELECT venue\_id,venue\_name,

(SELECT AVG(ticket\_price) FROM event e

WHERE e.venue\_id=v.venue\_id) AS avg\_ticket\_price

FROM venue v;

