**JAVA FOUNDATION TRAINING**

**BATCH 8**

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

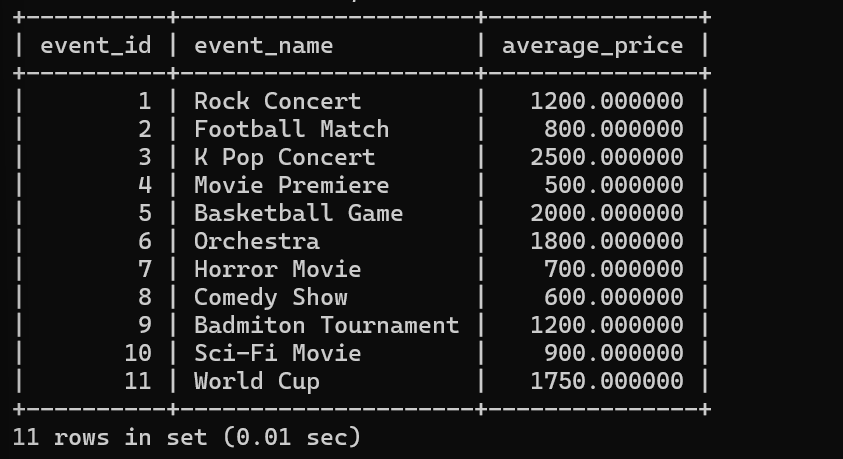
**Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:**

**1. Write a SQL query to List Events and Their Average Ticket Prices.**

**🡪** SELECT event\_id, event\_name, AVG (ticket\_price) AS average\_price

FROM event

GROUP BY event\_id;

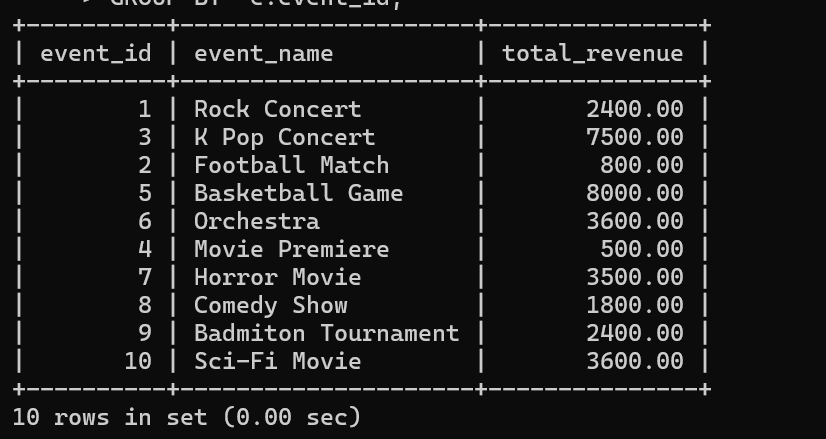


**2. Write a SQL query to Calculate the Total Revenue Generated by Events.**

**🡪** SELECT e.event\_id, e.event\_name, SUM (b.total\_cost) AS total\_revenue

FROM event e, booking b WHERE e.event\_id=b.event\_id

GROUP BY e.event\_id;

****

**3. Write a SQL query to find the event with the highest ticket sales.**

**🡪** SELECT e.event\_id, e.event\_name, SUM(b.num\_tickets) AS ticket\_sale

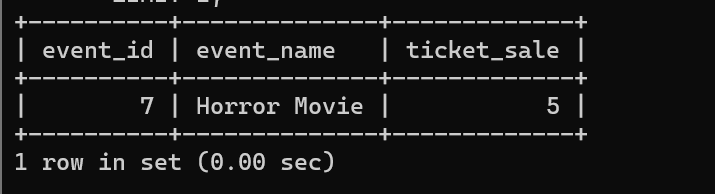
FROM booking b , event e

WHERE b.event\_id = e.event\_id

GROUP BY e.event\_id, e.event\_name

ORDER BY ticket\_sale DESC

LIMIT 1;



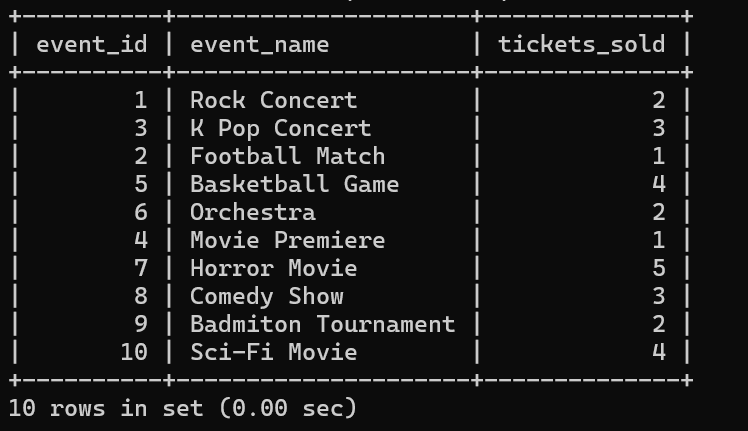
**4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.**

**🡪** SELECT e.event\_id, e.event\_name, SUM(b.num\_tickets) AS tickets\_sold

FROM event e, booking b

WHERE e.event\_id=b.event\_id

GROUP BY event\_id,event\_name;



**5. Write a SQL query to Find Events with No Ticket Sales.**

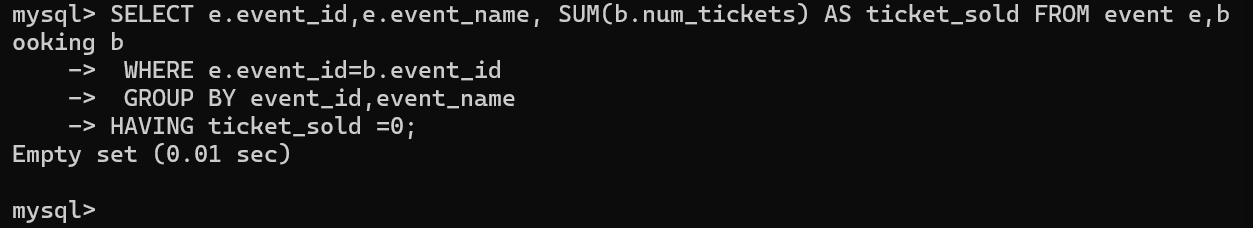
**🡪** SELECT e.event\_id,e.event\_name, SUM(b.num\_tickets) AS ticket\_sold

FROM event e,booking b

WHERE e.event\_id=b.event\_id

GROUP BY event\_id,event\_name

HAVING ticket\_sold =0;



**6. Write a SQL query to Find the User Who Has Booked the Most Tickets.**

**🡪** SELECT c.customer\_id,

c.customer\_name,e.event\_id,e.event\_name,

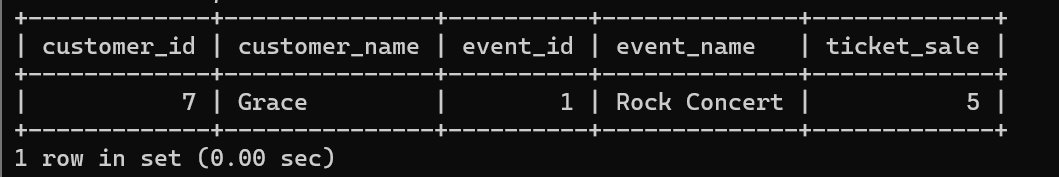
SUM(b.num\_tickets) AS ticket\_sale

FROM customer c,booking b,event e

WHERE c.customer\_id=b.customer\_id

GROUP BY c.customer\_id,e.event\_id  
ORDER BY ticket\_sale DESC

LIMIT 1;

****

**7. Write a SQL query to List Events and the total number of tickets sold for each month.**

**🡪** SELECT e.event\_id,e.event\_name, MONTH(e.event\_date)

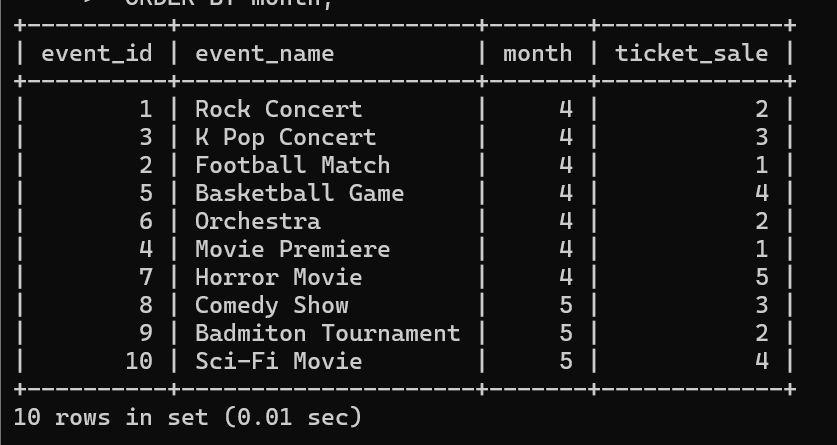
AS month,SUM(b.num\_tickets) AS ticket\_sale

FROM event e,booking b

WHERE e.event\_id=b.event\_id

GROUP BY e.event\_id

ORDER BY month;

****

**8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.**

🡪 SELECT v.venue\_id,v.venue\_name, AVG(e.ticket\_price) AS avg\_price

FROM event e,venue v

WHERE e.venue\_id=v.venue\_id  
 GROUP BY v.venue\_id;

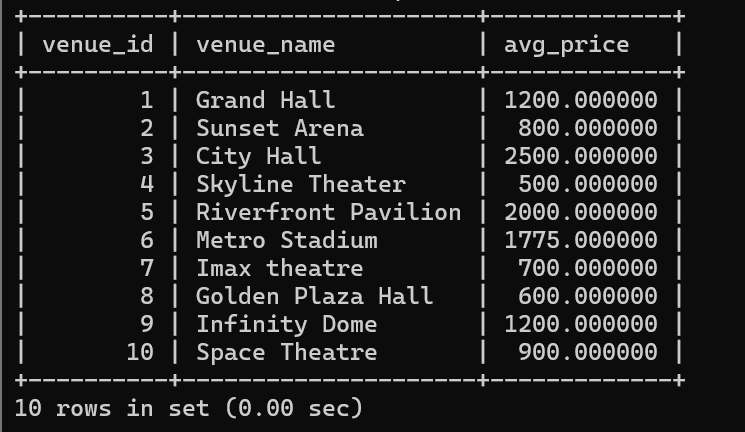
**Using JOINS:**

🡪 SELECT v.venue\_id, v.venue\_name, AVG(e.ticket\_price) AS avg\_price

FROM event e

JOIN venue v ON e.venue\_id = v.venue\_id

GROUP BY v.venue\_id,

****

**9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.**

**🡪** SELECT e.event\_type,SUM(b.num\_tickets) AS

total\_tickets\_sold

FROM event e,booking b

WHERE e.event\_id=b.event\_id

GROUP BY e.event\_type;

**Using JOINS:**

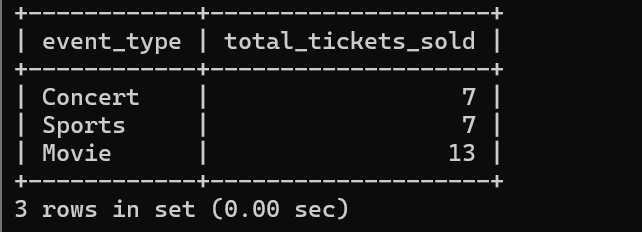
**🡪** SELECT e.event\_type, SUM(b.num\_tickets) AS

total\_tickets\_sold

FROM booking b

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

GROUP BY e.event\_type;

****

**10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.**

**🡪** SELECT e.event\_id,e.event\_name,YEAR(e.event\_date) AS

year, SUM(b.total\_cost)

FROM event e,booking b

WHERE e.event\_id=b.event\_id

GROUP BY e.event\_id

ORDER BY year;

**Using JOINS:**

**🡪** SELECT e.event\_id,e.event\_name,YEAR(e.event\_date) AS

year, SUM(b.total\_cost)

FROM event e

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

GROUP BY e.event\_id

ORDER BY year;

****

**11. Write a SQL query to list users who have booked tickets for multiple events.**

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

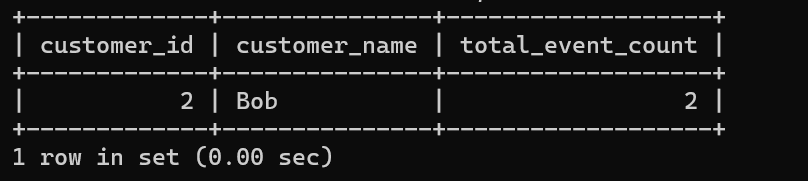
COUNT(DISTINCT b.event\_id) AS total\_event\_count

FROM customer c,booking b

WHERE c.customer\_id=b.customer\_id

GROUP BY c.customer\_id

HAVING total\_event\_count>1;

****

**Using JOINS:**

**🡪** SELECT c.customer\_id, c.customer\_name, COUNT(DISTINCT b.event\_id) AS total\_event\_count

FROM booking b

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

GROUP BY c.customer\_id, c.customer\_name

HAVING total\_event\_count > 1;

**12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.**

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

SUM(b.total\_cost) AS total\_revenue

FROM customer c,booking b

WHERE c.customer\_id=b.customer\_id

GROUP BY c.customer\_id;

**Using JOINS:**

**🡪** SELECT c.customer\_id, c.customer\_name, SUM(b.total\_cost) AS total\_revenue

FROM booking b

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

GROUP BY c.customer\_id, c.customer\_name;



**13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.**

**🡪** SELECT e.event\_type,v.venue\_name,AVG(e.ticket\_price) as

avg\_ticket\_price

FROM event e,venue v

WHERE e.venue\_id=v.venue\_id

GROUP BY e.event\_type,v.venue\_name;

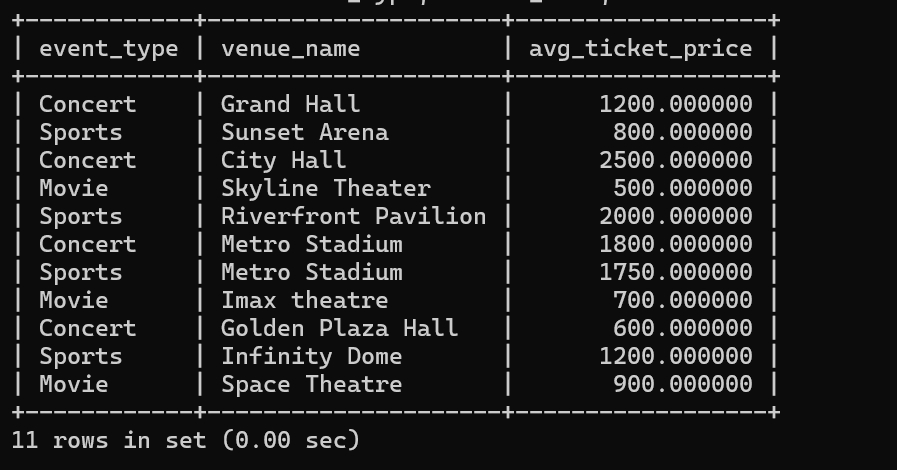
**Using JOINS:**

🡪 SELECT e.event\_type, v.venue\_name, AVG(e.ticket\_price) AS avg\_ticket\_price

FROM event e

JOIN venue v ON e.venue\_id = v.venue\_id

GROUP BY e.event\_type, v.venue\_name;



**14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.**

**🡪** SELECT c.customer\_id, c.customer\_name, SUM(b.num\_tickets) as total\_tickets

FROM customer c,booking b  
WHERE c.customer\_id=b.customer\_id AND

DATEDIFF(CURDATE(),b.booking\_date)<=30  
GROUP BY c.customer\_id;

**Using JOINS:**

**🡪** SELECT c.customer\_id, c.customer\_name, SUM(b.num\_tickets) AS total\_tickets

FROM booking b

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

WHERE DATEDIFF(CURDATE(), b.booking\_date) <= 30

GROUP BY c.customer\_id, c.customer\_name;

****