1)QUESTION

**User Upcoming Events**

Show a list of all upcoming events a user is registered for in their city, sorted by date.

QUERY

SELECT e.title, e.start\_date, e.city

FROM Events e

JOIN Registrations r ON e.event\_id = r.event\_id

JOIN Users u ON r.user\_id = u.user\_id

WHERE e.status = 'upcoming' AND u.city = e.city

ORDER BY e.start\_date;

OUTPUT

| **full\_name** | **title** | **city** | **start\_date** |
| --- | --- | --- | --- |
| Alice Johnson | Tech Innovators Meetup | New York | 2025-06-10 10:00:00 |
| Ethan Hunt | Frontend Development Bootcamp | Los Angeles | 2025-07-01 10:00:00 |

2)QUESTION

**Top Rated Events**

Identify events with the highest average rating, considering only those that have received at least 10 feedback submissions.

QUERY

SELECT e.title, AVG(f.rating) AS avg\_rating

FROM Events e

JOIN Feedback f ON e.event\_id = f.event\_id

GROUP BY e.event\_id

HAVING COUNT(f.feedback\_id) >= 10

ORDER BY avg\_rating DESC;

OUTPUT

Result: Empty

3)QUESTION

**Inactive Users**

Retrieve users who have not registered for any events in the last 90 days.

QUERY

SELECT u.user\_id, u.full\_name

FROM Users u

LEFT JOIN Registrations r ON u.user\_id = r.user\_id AND r.registration\_date >= CURDATE() - INTERVAL 90 DAY

WHERE r.registration\_id IS NULL;

OUTPUT

Result: Empty

4)QUESTION

**Peak session Hours**

Count how many sessions are scheduled between 10PM to 12 PM for each event

QUERY

SELECT e.title, COUNT(s.session\_id) AS session\_count

FROM Events e

JOIN Sessions s ON e.event\_id = s.event\_id

WHERE TIME(s.start\_time) BETWEEN '10:00:00' AND '12:00:00'

GROUP BY e.event\_id;

OUTPUT

| **event\_id** | **session\_count** | **title** |
| --- | --- | --- |
| 1 | 2 | Tech Innovators Meetup |
| 3 | 1 | Frontend Development Bootcamp |

5) QUESTION

**Most Active Cities**

List the top 5 cities with the highest number of distinct user registrations.

QUERY

SELECT e.city, COUNT(DISTINCT r.user\_id) AS user\_count

FROM Events e

JOIN Registrations r ON e.event\_id = r.event\_id

GROUP BY e.city

ORDER BY user\_count DESC

LIMIT 5;

OUTPUT

| **city** | **user\_count** |
| --- | --- |
| Los Angeles | 2 |
| New York | 2 |
| Chicago | 1 |

6)QUESTION

**Event Resource Summary**

Generate a report showing the number of resources (PDFs, images, links) uploaded for each event.

QUERY

SELECT e.title,

COUNT(CASE WHEN r.resource\_type = 'pdf' THEN 1 END) AS pdf\_count,

COUNT(CASE WHEN r.resource\_type = 'image' THEN 1 END) AS image\_count,

COUNT(CASE WHEN r.resource\_type = 'link' THEN 1 END) AS link\_count

FROM Events e

LEFT JOIN Resources r ON e.event\_id = r.event\_id

GROUP BY e.event\_id;

OUTPUT

| **event\_id** | **image** | **link** | **pdf** | **title** |
| --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | Tech Innovators Meetup |

7)QUESTION

**Low Feedback Alerts**

List all users who gave feedback with a rating less than 3, along with their comments and associated event names.

QUERY

SELECT u.full\_name, e.title, f.rating, f.comments

FROM Feedback f

JOIN Users u ON f.user\_id = u.user\_id

JOIN Events e ON f.event\_id = e.event\_id

WHERE f.rating < 3;

OUTPUT

Result: Empty

8)QUESTION

**Sessions per Upcoming Event**

Display all upcoming events with the count of sessions scheduled for them.

QUERY

SELECT e.title, COUNT(s.session\_id) AS session\_count

FROM Events e

LEFT JOIN Sessions s ON e.event\_id = s.event\_id

WHERE e.status = 'upcoming'

GROUP BY e.event\_id;

OUTPUT

| **event\_id** | **session\_count** | **title** |
| --- | --- | --- |
| 1 | 2 | Tech Innovators Meetup |
| 3 | 1 | Frontend Development Bootcamp |

9)QUESTION

**Organizer Event Summary**

For each event organizer, show the number of events created and their current status

(upcoming, completed, cancelled).

QUERY

SELECT u.full\_name,

COUNT(e.event\_id) AS total\_events,

SUM(e.status = 'upcoming') AS upcoming,

SUM(e.status = 'completed') AS completed,

SUM(e.status = 'cancelled') AS cancelled

FROM Users u

JOIN Events e ON u.user\_id = e.organizer\_id

GROUP BY u.user\_id;

OUTPUT

| **full\_name** | **status** | **event\_count** |
| --- | --- | --- |
| Alice Johnson | upcoming | 1 |
| Bob Smith | upcoming | 1 |
| Charlie Lee | completed | 1 |

10)QUESTION

**Feedback Gap**

Identify events that had registrations but received no feedback at all.

QUERY

SELECT e.title

FROM Events e

JOIN Registrations r ON e.event\_id = r.event\_id

LEFT JOIN Feedback f ON e.event\_id = f.event\_id

GROUP BY e.event\_id

HAVING COUNT(f.feedback\_id) = 0;

OUTPUT

Result empty

11)QUESTION

**Daily New User Count**

Find the number of users who registered each day in the last 7 days.

QUERY

SELECT registration\_date, COUNT(\*) AS new\_users

FROM Users

WHERE registration\_date >= CURDATE() - INTERVAL 7 DAY

GROUP BY registration\_date;

OUTPUT

Result empty

12)QUESTION

**Event with Maximum Sessions**

List the event(s) with the highest number of sessions.

QUERY

SELECT e.title, COUNT(s.session\_id) AS session\_count

FROM Events e

JOIN Sessions s ON e.event\_id = s.event\_id

GROUP BY e.event\_id

ORDER BY session\_count DESC

LIMIT 1;

OUTPUT

| **event\_id** | **session\_count** | **title** |
| --- | --- | --- |
| 1 | 2 | Tech Innovators Meetup |

13)QUESTION

**Average Rating per City**

Calculate the average feedback rating of events conducted in each city.

QUERY

OUTPUT

| **city** | **avg\_rating** |
| --- | --- |
| Chicago | 4.5 |
| New York | 3.0 |

14)QUESTION

**Most Registered Events**

List top 3 events based on the total number of user registrations.

QUERY

OUTPUT

| **event\_id** | **registration\_count** | **title** |
| --- | --- | --- |
| 1 | 2 | Tech Innovators Meetup |
| 2 | 2 | AI & ML Conference |
| 3 | 1 | Frontend Development Bootcamp |

15)QUESTION

**Event Session Time Conflict**

Identify overlapping sessions within the same event (i.e., session start and end times that conflict).

QUERY

SELECT s1.event\_id, s1.session\_id AS session1, s2.session\_id AS session2

FROM Sessions s1

JOIN Sessions s2 ON s1.event\_id = s2.event\_id AND s1.session\_id < s2.session\_id

WHERE s1.start\_time < s2.end\_time AND s2.start\_time < s1.end\_time;

OUTPUT

Result empty

16)QUESTION

**Unregistered Active Users**

Find users who created an account in the last 30 days but haven’t registered for any events.

QUERY

SELECT u.user\_id, u.full\_name

FROM Users u

LEFT JOIN Registrations r ON u.user\_id = r.user\_id

WHERE u.registration\_date >= CURDATE() - INTERVAL 30 DAY

AND r.registration\_id IS NULL;

OUTPUT

Result empty

17)QUESTION

**Multi-Session Speakers**

Identify speakers who are handling more than one session across all events.

QUERY

SELECT speaker\_name, COUNT(\*) AS session\_count

FROM Sessions

GROUP BY speaker\_name

HAVING COUNT(\*) > 1;

OUTPUT

Result empty

18)QUESTION

**Resource Availability Check**

List all events that do not have any resources uploaded.

QUERY

SELECT e.title

FROM Events e

LEFT JOIN Resources r ON e.event\_id = r.event\_id

GROUP BY e.event\_id

HAVING COUNT(r.resource\_id) = 0;

OUTPUT

Result empty

19)QUESTION

**Completed Events with Feedback Summary**

For completed events, show total registrations and average feedback rating.

QUERY

SELECT e.title, COUNT(DISTINCT r.user\_id) AS total\_registrations, AVG(f.rating) AS avg\_rating

FROM Events e

LEFT JOIN Registrations r ON e.event\_id = r.event\_id

LEFT JOIN Feedback f ON e.event\_id = f.event\_id

WHERE e.status = 'completed'

GROUP BY e.event\_id;

OUTPUT

| **title** | **total\_registrations** | **avg\_rating** |
| --- | --- | --- |
| AI & ML Conference | 2 | 4.5 |

20)QUESTION

**User Engagement Index**

For each user, calculate how many events they attended and how many feedbacks they submitted.

QUERY

SELECT u.user\_id, u.full\_name,

COUNT(DISTINCT r.event\_id) AS events\_attended,

COUNT(DISTINCT f.feedback\_id) AS feedbacks\_given

FROM Users u

LEFT JOIN Registrations r ON u.user\_id = r.user\_id

LEFT JOIN Feedback f ON u.user\_id = f.user\_id

GROUP BY u.user\_id;

OUTPUT

| **user\_id** | **full\_name** | **events\_attended** | **feedbacks\_given** |
| --- | --- | --- | --- |
| 1 | Alice Johnson | 1 | 0 |
| 2 | Bob Smith | 1 | 1 |
| 3 | Charlie Lee | 1 | 1 |
| 4 | Diana King | 1 | 1 |
| 5 | Ethan Hunt | 1 | 0 |

21)QUESTION

**Top Feedback Providers**

List top 5 users who have submitted the most feedback entries.

QUERY

SELECT u.full\_name, COUNT(f.feedback\_id) AS feedback\_count

FROM Feedback f

JOIN Users u ON f.user\_id = u.user\_id

GROUP BY f.user\_id

ORDER BY feedback\_count DESC

LIMIT 5;

OUTPUT

| **full\_name** | **feedback\_count** |
| --- | --- |
| Charlie Lee | 1 |
| Diana King | 1 |
| Bob Smith | 1 |

22)QUESTION

**Duplicate Registrations Check**

Detect if a user has been registered more than once for the same event.

QUERY

SELECT user\_id, event\_id, COUNT(\*) AS duplicate\_count

FROM Registrations

GROUP BY user\_id, event\_id

HAVING COUNT(\*) > 1;

OUTPUT

Result empty

23)QUESTION

**Registration Trends**

Show a month-wise registration count trend over the past 12 months.

QUERY

SELECT DATE\_FORMAT(registration\_date, '%Y-%m') AS month, COUNT(\*) AS total\_registrations

FROM Registrations

WHERE registration\_date >= CURDATE() - INTERVAL 12 MONTH

GROUP BY month

ORDER BY month;

OUTPUT

| **month** | **total\_registrations** |
| --- | --- |
| 2025-04 | 2 |
| 2025-05 | 2 |
| 2025-06 | 1 |

24)QUESTION

**Average Session Duration per Event**

Compute the average duration (in minutes) of sessions in each event.

QUERY

SELECT e.title,

AVG(TIMESTAMPDIFF(MINUTE, s.start\_time, s.end\_time)) AS avg\_duration\_mins

FROM Events e

JOIN Sessions s ON e.event\_id = s.event\_id

GROUP BY e.event\_id;

OUTPUT

| **event\_id** | **title** | **avg\_duration\_minutes** |
| --- | --- | --- |
| 1 | Tech Innovators Meetup | 67.5 |
| 2 | AI & ML Conference | 90.0 |
| 3 | Frontend Development Bootcamp | 120.0 |

25)QUESTION

**Events Without Sessions**

List all events that currently have no sessions scheduled under them.

QUERY

SELECT e.title

FROM Events e

LEFT JOIN Sessions s ON e.event\_id = s.event\_id

GROUP BY e.event\_id

HAVING COUNT(s.session\_id) = 0;

OUTPUT

Result empty