ANSI SQL Using MySQL - Exercise Solutions

1. User Upcoming Events

```
SELECT u.full_name, e.title, e.start_date
FROM Users u
JOIN Registrations r ON u.user_id = r.user_id
JOIN Events e ON r.event_id = e.event_id
WHERE e.status = 'upcoming' AND u.city = e.city
ORDER BY e.start_date;
```

2. Top Rated Events

```
SELECT e.title, AVG(f.rating) AS avg_rating, COUNT(f.feedback_id) AS
feedback_count
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;
```

3. Inactive Users

```
SELECT *
FROM Users
WHERE user_id NOT IN (
   SELECT user_id
   FROM Registrations
   WHERE registration_date >= CURDATE() - INTERVAL 90 DAY
);
```

4. Peak Session Hours

```
SELECT event_id, COUNT(*) AS session_count
FROM Sessions
WHERE TIME(start_time) BETWEEN '10:00:00' AND '11:59:59'
GROUP BY event_id;
```

5. Most Active Cities

```
SELECT u.city, COUNT(DISTINCT r.user_id) AS total_users
FROM Users u
JOIN Registrations r ON u.user_id = r.user_id
GROUP BY u.city
```

```
ORDER BY total_users DESC
LIMIT 5;
```

6. Event Resource Summary

```
SELECT event_id,
   SUM(resource_type = 'pdf') AS pdf_count,
   SUM(resource_type = 'image') AS image_count,
   SUM(resource_type = 'link') AS link_count
FROM Resources
GROUP BY event_id;
```

7. Low Feedback Alerts

```
SELECT u.full_name, e.title AS event_name, 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;
```

8. Sessions per Upcoming Event

```
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;
```

9. Organizer Event Summary

```
SELECT u.full_name AS organizer, e.status, COUNT(e.event_id) AS event_count
FROM Events e
JOIN Users u ON e.organizer_id = u.user_id
GROUP BY u.full_name, e.status;
```

10. Feedback Gap

```
SELECT DISTINCT e.title
FROM Events e
JOIN Registrations r ON e.event_id = r.event_id
WHERE e.event_id NOT IN (
   SELECT DISTINCT event_id FROM Feedback
);
```

11. Daily New User Count

```
SELECT registration_date, COUNT(*) AS user_count
FROM Users
```

```
WHERE registration_date >= CURDATE() - INTERVAL 7 DAY
GROUP BY registration_date;
```

12. Event with Maximum Sessions

```
SELECT e.title, COUNT(s.session_id) AS total_sessions
FROM Events e
JOIN Sessions s ON e.event_id = s.event_id
GROUP BY e.event_id
ORDER BY total_sessions DESC
LIMIT 1;
```

13. Average Rating per City

```
SELECT e.city, AVG(f.rating) AS avg_rating
FROM Feedback f
JOIN Events e ON f.event_id = e.event_id
GROUP BY e.city;
```

14. Most Registered Events

```
SELECT e.title, COUNT(r.registration_id) AS total_registrations
FROM Events e
JOIN Registrations r ON e.event_id = r.event_id
GROUP BY e.event_id
ORDER BY total_registrations DESC
LIMIT 3;
```

15. Event Session Time Conflict

```
SELECT s1.event_id, s1.title AS session1, s2.title 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 s1.end_time > s2.start_time;
```

16. Unregistered Active Users

```
SELECT *
FROM Users
WHERE registration_date >= CURDATE() - INTERVAL 30 DAY
AND user_id NOT IN (
   SELECT DISTINCT user_id FROM Registrations
);
```

17. Multi-Session Speakers

```
SELECT speaker_name, COUNT(*) AS session_count
FROM Sessions
```

```
GROUP BY speaker_name
HAVING COUNT(*) > 1;
```

18. Events Without Resources

```
SELECT title
FROM Events
WHERE event_id NOT IN (
   SELECT DISTINCT event_id FROM Resources
);
```

19. Completed Events with Feedback Summary

```
SELECT e.title, COUNT(DISTINCT r.registration_id) AS total_regs,
      ROUND(AVG(f.rating), 2) 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;
```

20. User Engagement Index

```
SELECT 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;
```

21. Top Feedback Providers

```
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;
```

22. Duplicate Registrations Check

```
SELECT user_id, event_id, COUNT(*) AS reg_count
FROM Registrations
GROUP BY user_id, event_id
HAVING COUNT(*) > 1;
```

23. Registration Trends

```
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;
```

24. Average Session Duration per Event

```
SELECT event_id,
    ROUND(AVG(TIMESTAMPDIFF(MINUTE, start_time, end_time)), 2) AS
avg_duration_minutes
FROM Sessions
GROUP BY event_id;
```

25. Events Without Sessions

```
SELECT e.title
FROM Events e
LEFT JOIN Sessions s ON e.event_id = s.event_id
WHERE s.session_id IS NULL;
```