ANSI SQL Using MySQL Exercises Solution

- Creating table according to the schema

  1. Users

     Code:

     ```
     CREATE TABLE Users (
        user_id INT PRIMARY KEY AUTO_INCREMENT,
        full_name VARCHAR(100) NOT NULL,
        email VARCHAR(100) UNIQUE NOT NULL,
        city VARCHAR(100) NOT NULL,
        registration_date DATE NOT NULL
     );
     ```

  2. Events

     Code:

     ```
     CREATE TABLE Events (
        event_id INT PRIMARY KEY AUTO_INCREMENT,
        title VARCHAR(200) NOT NULL,
        description TEXT,
        city VARCHAR(100) NOT NULL,
        start_date DATETIME NOT NULL,
        end_date DATETIME NOT NULL,
        status ENUM('upcoming', 'completed', 'cancelled'),
        organizer_id INT,
        FOREIGN KEY (organizer_id) REFERENCES
     Users(user_id)
     );
     ```

  3. Sessions

     Code:

     ```
     CREATE TABLE Sessions (
        session_id INT PRIMARY KEY AUTO_INCREMENT,
        event_id INT,
     ```

```
    title VARCHAR(200) NOT NULL,

    speaker_name VARCHAR(100) NOT NULL,

    start_time DATETIME NOT NULL,

    end_time DATETIME NOT NULL,

    FOREIGN KEY (event_id) REFERENCES
Events(event_id)

);
```

4. <u>Registrations</u>

    <u>Code:</u>

```
CREATE TABLE Registrations (
   registration_id INT PRIMARY KEY
AUTO_INCREMENT,
   user_id INT,
   event_id INT,
   registration_date DATE NOT NULL,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (event_id) REFERENCES
Events(event_id)
);
```

5. <u>Feedback</u>

    <u>Code:</u>

```
CREATE TABLE Feedback (
   feedback_id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT,
   event_id INT,
   rating INT CHECK (rating BETWEEN 1 AND 5),
   comments TEXT,
   feedback_date DATE NOT NULL,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (event_id) REFERENCES
Events(event_id)
);
```

6. <u>Resources</u>

    <u>Code:</u>

```
CREATE TABLE Resources (
   resource_id INT PRIMARY KEY AUTO_INCREMENT,
    event_id INT,
    resource_type ENUM('pdf', 'image', 'link'),
    resource_url VARCHAR(255) NOT NULL,
    uploaded_at DATETIME NOT NULL,
    FOREIGN KEY (event_id) REFERENCES
Events(event_id)
);
```

➢ Inserting data in the table according to the sample dataset

- <u>Users:</u>

  <u>Code:</u>

  ```
  INSERT INTO Users (user_id, full_name, email, city,
  registration_date) VALUES
  (1, 'Alice Johnson', 'alice@example.com', 'New York',
  '2024-12-01'),
  (2, 'Bob Smith', 'bob@example.com', 'Los Angeles',
  '2024-12-05'),
  (3, 'Charlie Lee', 'charlie@example.com', 'Chicago',
  '2024-12-10'),
  (4, 'Diana King', 'diana@example.com', 'New York',
  '2025-01-15'),
  (5, 'Ethan Hunt', 'ethan@example.com', 'Los Angeles',
  '2025-02-01');
  ```

- <u>Events</u>

  <u>Code:</u>

  ```
  INSERT INTO Events (event_id, title, description,
  city, start_date, end_date, status, organizer_id)
  VALUES
  (1, 'Tech Innovators Meetup', 'A meetup for tech
  enthusiasts.', 'New York', '2025-06-10 10:00:00',
  '2025-06-10 16:00:00', 'upcoming', 1),
  (2, 'AI & ML Conference', 'Conference on AI and ML
  advancements.', 'Chicago', '2025-05-15 09:00:00',
  '2025-05-15 17:00:00', 'completed', 3),
  (3, 'Frontend Development Bootcamp', 'Hands-on
  training on frontend tech.', 'Los Angeles', '2025-07-01
  10:00:00', '2025-07-03 16:00:00', 'upcoming', 2);
  ```

- <u>Sessions</u>

  <u>Code:</u>

```
INSERT INTO Sessions (session_id, event_id, title,
speaker_name, start_time, end_time) VALUES
(1, 1, 'Opening Keynote', 'Dr. Tech', '2025-06-10
10:00:00', '2025-06-10 11:00:00'),
(2, 1, 'Future of Web Dev', 'Alice Johnson', '2025-06-
10 11:15:00', '2025-06-10 12:30:00'),
(3, 2, 'AI in Healthcare', 'Charlie Lee', '2025-05-15
09:30:00', '2025-05-15 11:00:00'),
(4, 3, 'Intro to HTML5', 'Bob Smith', '2025-07-01
10:00:00', '2025-07-01 12:00:00');
```

- Registrations

  Code:

  ```
  INSERT INTO Registrations (registration_id, user_id,
  event_id, registration_date) VALUES
  (1, 1, 1, '2025-05-01'),
  (2, 2, 1, '2025-05-02'),
  (3, 3, 2, '2025-04-30'),
  (4, 4, 2, '2025-04-28'),
  (5, 5, 3, '2025-06-15');
  ```

- Feedback

  Code:

  ```
  INSERT INTO Feedback (feedback_id, user_id,
  event_id, rating, comments, feedback_date) VALUES
  (1, 3, 2, 4, 'Great insights!', '2025-05-16'),
  (2, 4, 2, 5, 'Very informative.', '2025-05-16'),
  (3, 2, 1, 3, 'Could be better.', '2025-06-11');
  ```

- Resources

  Code:

  ```
  INSERT INTO Resources (resource_id, event_id,
  resource_type, resource_url, uploaded_at) VALUES
  (1, 1, 'pdf',
  'https://portal.com/resources/tech_meetup_agenda.pdf'
  , '2025-05-01 10:00:00'),
  (2, 2, 'image',
  'https://portal.com/resources/ai_poster.jpg', '2025-04-
  20 09:00:00'),
  (3, 3, 'link', 'https://portal.com/resources/html5_docs',
  '2025-06-25 15:00:00');
  ```

❖ EXERCISE:

1) User Upcoming Events:

```
SELECT u.full_name, e.title, e.city, 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 e.city = u.city
ORDER BY e.start_date;
```

2) Top Rated Events:

```
SELECT e.title, AVG(f.rating) AS avg_rating
FROM Feedback f
JOIN Events e ON f.event_id = e.event_id
GROUP BY e.event_id, e.title
HAVING COUNT(f.feedback_id) >= 10
ORDER BY avg_rating DESC;
```

3) Inactive Users:

```
SELECT u.*
FROM Users u
LEFT JOIN Registrations r ON u.user_id = r.user_id
GROUP BY u.user_id
HAVING MAX(r.registration_date) IS NULL
   OR MAX(r.registration_date) < DATE('now', '-90 days');
```

4) Peak Session Hours:

```
SELECT e.title, COUNT(*) AS session_count
FROM Sessions s
JOIN Events e ON s.event_id = e.event_id
WHERE TIME(s.start_time) BETWEEN '10:00:00' AND
'11:59:59'
GROUP BY e.event_id, e.title;
```

5) Most Active Cities:

```
SELECT u.city, COUNT(DISTINCT r.user_id) AS user_count
FROM Users u
```

```
JOIN Registrations r ON u.user_id = r.user_id
GROUP BY u.city
ORDER BY user_count DESC
LIMIT 5;
```

6) Event Resource Summary:

```
SELECT e.title,
     SUM(CASE WHEN r.resource_type = 'pdf' THEN 1 ELSE 0
END) AS pdfs,
     SUM(CASE WHEN r.resource_type = 'image' THEN 1 ELSE
0 END) AS images,
     SUM(CASE WHEN r.resource_type = 'link' THEN 1 ELSE 0
END) AS links
FROM Events e
LEFT JOIN Resources r ON e.event_id = r.event_id
GROUP BY e.title;
```

7) Low Feedback Alerts:

```
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;
```

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, e.title;
```

9) Organizer Event Summary:

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

10) Feedback Gap:

```
SELECT e.title
FROM Events e
JOIN Registrations r ON e.event_id = r.event_id
LEFT JOIN Feedback f ON r.event_id = f.event_id
GROUP BY e.event_id, e.title
HAVING COUNT(f.feedback_id) = 0;
```

11) Daily New User Count:

```
SELECT registration_date, COUNT(*) AS new_users
FROM Users
WHERE registration_date >= DATE('now', '-6 days')
GROUP BY registration_date
ORDER BY registration_date;
```

12) Event with Maximum Sessions:

```
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, e.title
ORDER BY session_count DESC
LIMIT 1;
```

13) Average Rating per City:

```
SELECT e.city, AVG(f.rating) AS avg_rating
FROM Events e
JOIN Feedback f ON e.event_id = f.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, e.title
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 s2.start_time <
s1.end_time;
```

16) Unregistered Active Users:

```
SELECT u.*
FROM Users u
LEFT JOIN Registrations r ON u.user_id = r.user_id
WHERE u.registration_date >= DATE('now', '-30 days')
  AND r.registration_id IS NULL;
```

17) Multi-Session Speakers:

```
SELECT speaker_name, COUNT(*) AS session_count
FROM Sessions
GROUP BY speaker_name
HAVING COUNT(*) > 1;
```

18) Resource Availability Check:

```
SELECT e.title
FROM Events e
LEFT JOIN Resources r ON e.event_id = r.event_id
GROUP BY e.event_id, e.title
HAVING COUNT(r.resource_id) = 0;
```

19) Completed Events with Feedback Summary:

```
SELECT e.title, COUNT(DISTINCT r.user_id) AS
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, e.title;
```

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 Users u
JOIN Feedback f ON u.user_id = f.user_id
GROUP BY u.user_id
ORDER BY feedback_count DESC
LIMIT 5;
```

22) Duplicate Registrations Check:

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

23) Registration Trends:

```
SELECT STRFTIME('%Y-%m', registration_date) AS
month, COUNT(*) AS registrations
FROM Registrations
GROUP BY month
ORDER BY month;
```

24) Average Session Duration per Event:

```
SELECT e.title,
      AVG(
         (JULIANDAY(s.end_time) -
JULIANDAY(s.start_time)) * 24 * 60
      ) AS avg_duration_minutes
```

```
FROM Events e
JOIN Sessions s ON e.event_id = s.event_id
GROUP BY e.event_id, e.title;
```

25) Events Without Sessions:

```
SELECT e.title
FROM Events e
LEFT JOIN Sessions s ON e.event_id = s.event_id
GROUP BY e.event_id, e.title
HAVING COUNT(s.session_id) = 0;
```