Volleyball Tournament Database

Goal: Manage tournament results and player stats.

Tables:

- `teams(team_id, name, region)`
- `matches(match_id, team1_id, team2_id, score_team1, score_team2, date)`
- `players(player_id, team_id, name, position)`

Queries:

- Find teams with the most wins.
- Identify top scorers across matches.
- Track regional performance trends.

a. Create database

```
CREATE DATABASE volleyball_tournament;
USE volleyball_tournament;
```

```
mysql> CREATE DATABASE volleyball_tournament;
Query OK, 1 row affected (0.05 sec)

mysql> USE volleyball_tournament;
Database changed
```

b. Create tables

```
-- Team information
CREATE TABLE teams (
    team_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    region VARCHAR(50)
);

-- Player information
CREATE TABLE players (
    player_id INT AUTO_INCREMENT PRIMARY KEY,
    team_id INT,
    name VARCHAR(100),
    position VARCHAR(30),
    FOREIGN KEY (team_id) REFERENCES teams(team_id)
);

-- Match results
CREATE TABLE matches (
    match_id INT AUTO_INCREMENT PRIMARY KEY,
    team1_id INT,
    team2_id INT,
    score_team1 INT,
    score_team2 INT,
    date DATE,
    FOREIGN KEY (team1_id) REFERENCES teams(team_id),
    FOREIGN KEY (team2_id) REFERENCES teams(team_id)
);
```

```
mysql> -- Team information
mysql> CREATE TABLE teams (
    ->     team_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100),
    ->     region VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql>
mysql> -- Player information
mysql> CREATE TABLE players (
    ->     player_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     team_id INT,
    ->     name VARCHAR(100),
    ->     position VARCHAR(30),
    ->     FOREIGN KEY (team_id) REFERENCES teams(team_id)
    -> );
Query OK, 0 rows affected (0.22 sec)

mysql>
mysql> -- Match results
mysql> CREATE TABLE matches (
    ->     match_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     team1_id INT,
    ->     team2_id INT,
    ->     score_team1 INT,
    ->     score_team2 INT,
    ->     date DATE,
    ->     FOREIGN KEY (team1_id) REFERENCES teams(team_id),
    ->     FOREIGN KEY (team2_id) REFERENCES teams(team_id)
    -> );
Query OK, 0 rows affected (0.19 sec)
```

c. Insert sample data

```
INSERT INTO teams (name, region)
VALUES
('Jakarta Spikers', 'Jakarta'),
('Bandung Smashers', 'Bandung'),
('Surabaya Blockers', 'Surabaya');

INSERT INTO players (team_id, name, position)
VALUES
(1, 'Alice Johnson', 'Outside Hitter'),
(1, 'Bob Smith', 'Setter'),
(2, 'Charlie Lee', 'Middle Blocker'),
(3, 'Diana Kim', 'Libero');

INSERT INTO matches (team1_id, team2_id, score_team1, score_team2,
date)
VALUES
(1, 2, 3, 1, '2025-11-01'),
(2, 3, 2, 3, '2025-11-03'),
(1, 3, 3, 2, '2025-11-05');
```

```
mysql> INSERT INTO teams (name, region)
    -> VALUES
    -> ('Jakarta Spikers', 'Jakarta'),
    -> ('Bandung Smashers', 'Bandung'),
    -> ('Surabaya Blockers', 'Surabaya');
Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO players (team_id, name, position)
    -> VALUES
    -> (1, 'Alice Johnson', 'Outside Hitter'),
    -> (1, 'Bob Smith', 'Setter'),
    -> (2, 'Charlie Lee', 'Middle Blocker'),
    -> (3, 'Diana Kim', 'Libero');
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO matches (team1_id, team2_id, score_team1, score_team2, date)
    -> VALUES
    -> (1, 2, 3, 1, '2025-11-01'),
    -> (2, 3, 2, 3, '2025-11-03'),
    -> (1, 3, 3, 2, '2025-11-05');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

d. Example queries

```
-- Find teams with the most wins
SELECT t.name, COUNT(*) AS wins
FROM teams t
JOIN matches m ON t.team_id = m.team1_id
WHERE m.score_team1 > m.score_team2
GROUP BY t.name
UNION
SELECT t.name, COUNT(*) AS wins
FROM teams t
JOIN matches m ON t.team_id = m.team2_id
WHERE m.score_team2 > m.score_team1
GROUP BY t.name;

-- Identify top scorers across matches (example: points from
score_team1/score_team2)
SELECT t.name, SUM(CASE WHEN t.team_id = m.team1_id THEN
m.score_team1
                        WHEN t.team_id = m.team2_id THEN
m.score_team2 END) AS total_points
FROM teams t
JOIN matches m ON t.team_id IN (m.team1_id, m.team2_id)
GROUP BY t.name
ORDER BY total_points DESC;

-- Track regional performance trends
SELECT region, COUNT(*) AS matches_played
FROM teams t
JOIN matches m ON t.team_id IN (m.team1_id, m.team2_id)
GROUP BY region;
```

```
mysql> -- Find teams with the most wins
mysql> SELECT t.name, COUNT(*) AS wins
    -> FROM teams t
    -> JOIN matches m ON t.team_id = m.team1_id
    -> WHERE m.score_team1 > m.score_team2
    -> GROUP BY t.name
    -> UNION
    -> SELECT t.name, COUNT(*) AS wins
    -> FROM teams t
    -> JOIN matches m ON t.team_id = m.team2_id
    -> WHERE m.score_team2 > m.score_team1
    -> GROUP BY t.name;
+------------------+------+
| name             | wins |
+------------------+------+
| Jakarta Spikers  |    2 |
| Surabaya Blockers |    1 |
+------------------+------+
2 rows in set (0.04 sec)

mysql>
mysql> -- Identify top scorers across matches (example: points from score_team1/score_team2)
mysql> SELECT t.name, SUM(CASE WHEN t.team_id = m.team1_id THEN m.score_team1
    ->                         WHEN t.team_id = m.team2_id THEN m.score_team2 END) AS total_points
    -> FROM teams t
    -> JOIN matches m ON t.team_id IN (m.team1_id, m.team2_id)
    -> GROUP BY t.name
    -> ORDER BY total_points DESC;
+------------------+--------------+
| name             | total_points |
+------------------+--------------+
| Jakarta Spikers  |            6 |
| Surabaya Blockers |            5 |
| Bandung Smashers |            3 |
+------------------+--------------+
3 rows in set (0.00 sec)
```

```
mysql>
mysql> -- Track regional performance trends
mysql> SELECT region, COUNT(*) AS matches_played
    -> FROM teams t
    -> JOIN matches m ON t.team_id IN (m.team1_id, m.team2_id)
    -> GROUP BY region;
+----------+----------------+
| region   | matches_played |
+----------+----------------+
| Jakarta  |              2 |
| Bandung  |              2 |
| Surabaya |              2 |
+----------+----------------+
3 rows in set (0.00 sec)
```