**Sports League Analysis Question Set and answers**

**-- 1. Query to calculate the total points scored by each player**

SELECT

p.player\_name,

SUM(ps.points) AS total\_points

FROM

PlayerStats AS ps

JOIN

Players AS p ON ps.player\_id = p.player\_id

GROUP BY

p.player\_name

ORDER BY

total\_points DESC;

**-- 2. Query to find players who scored points between 3 and 6**

SELECT

p.player\_name,

ps.points

FROM

PlayerStats AS ps

JOIN

Players AS p ON ps.player\_id = p.player\_id

WHERE

ps.points BETWEEN 3 AND 6

ORDER BY

ps.points ASC;

**-- 3. Find players from the same team**

-- This query finds pairs of players on the same team.

SELECT

p1.player\_id AS player1\_id,

p1.player\_name AS player1\_name,

p2.player\_id AS player2\_id,

p2.player\_name AS player2\_name,

t.team\_name

FROM Players p1

JOIN Players p2

ON p1.team\_id = p2.team\_id

AND p1.player\_id < p2.player\_id -- avoids duplicates & self-pairing

JOIN Teams t

ON p1.team\_id = t.team\_id

WHERE p1.team\_id IS NOT NULL

ORDER BY t.team\_name, p1.player\_name, p2.player\_name;

**-- 4. Find games played in the last 30 days**

SELECT \*

FROM

Games

WHERE

game\_date >= DATE\_SUB(CURDATE(), INTERVAL 30 DAY)

AND game\_date <= CURDATE();

**-- 5. Create a view to summarize player statistics**

CREATE OR REPLACE VIEW PlayerStatisticsSummary AS

SELECT

p.player\_name,

t.team\_name,

COUNT(ps.game\_id) AS games\_played,

COALESCE(SUM(ps.points), 0) AS total\_points,

COALESCE(AVG(ps.points), 0) AS avg\_points\_per\_game,

COALESCE(SUM(ps.assists), 0) AS total\_assists,

COALESCE(SUM(ps.rebounds), 0) AS total\_rebounds

FROM Players p

LEFT JOIN Teams t ON p.team\_id = t.team\_id

LEFT JOIN PlayerStats ps ON p.player\_id = ps.player\_id

GROUP BY p.player\_id, p.player\_name, t.team\_name;

-- After creating the view, query it like this:

SELECT \* FROM PlayerStatisticsSummary;

**-- 6. Create a trigger to ensure points cannot be negative before inserting or updating**

-- This trigger is set up to run BEFORE an INSERT or UPDATE on the PlayerStats table.

-- If the new 'points' value is less than 0, it will raise an error and prevent the operation.

DELIMITER $$

CREATE TRIGGER check\_negative\_points

BEFORE INSERT ON PlayerStats

FOR EACH ROW

BEGIN

IF NEW.points < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Points cannot be a negative value.';

END IF;

END$$

DELIMITER ;

-- You can also create a separate trigger for updates

DELIMITER $$

CREATE TRIGGER check\_negative\_points\_update

BEFORE UPDATE ON PlayerStats

FOR EACH ROW

BEGIN

IF NEW.points < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Points cannot be a negative value.';

END IF;

END$$

DELIMITER ;

**-- 7. Fetch all players and their respective teams, including players without a team**

SELECT

p.player\_name,

t.team\_name

FROM

Players AS p

LEFT JOIN

Teams AS t ON p.team\_id = t.team\_id;

**-- 8. Total points scored by players, grouped by their teams**

SELECT

t.team\_name,

SUM(ps.points) AS total\_team\_points

FROM

Teams AS t

JOIN

Players AS p ON t.team\_id = p.team\_id

JOIN

PlayerStats AS ps ON p.player\_id = ps.player\_id

GROUP BY

t.team\_name

ORDER BY

total\_team\_points DESC;

**-- 9. Players who scored more than 5 points**

SELECT

p.player\_name,

ps.points

FROM

PlayerStats AS ps

JOIN

Players AS p ON ps.player\_id = p.player\_id

WHERE

ps.points > 5

ORDER BY

ps.points DESC;

**-- 10. Update and assign Sarah Moore to the team Green Sharks**

set sql\_safe\_updates=0;

UPDATE Players

SET team\_id = (SELECT team\_id FROM Teams WHERE team\_name = 'Green Sharks')

WHERE player\_name = 'Sarah Moore';

**-- 11. Deleting all records where the game id is 5**

DELETE FROM PlayerStats WHERE game\_id = 5;

DELETE FROM Games WHERE game\_id = 5;

**-- 12. Players who scored more than the average points in a specific game**

-- This query finds players who scored more than the average points of the game they participated in.

SELECT

p.player\_name,

ps.points,

ps.game\_id

FROM

PlayerStats AS ps

JOIN

Players AS p ON ps.player\_id = p.player\_id

WHERE

ps.points > (SELECT AVG(points) FROM PlayerStats WHERE game\_id = ps.game\_id);

**-- 13. Find the top 3 players who have scored the highest total points across all games.**

SELECT

p.player\_name,

SUM(ps.points) AS total\_points

FROM

PlayerStats AS ps

JOIN

Players AS p ON ps.player\_id = p.player\_id

GROUP BY

p.player\_name

ORDER BY

total\_points DESC

LIMIT 3;

**-- 14. Retrieve a list of teams that have won at least one game, considering a win as having a higher score than the opposing team.**

SELECT distinct

g.game\_id,

t.team\_name AS winning\_team,

GREATEST(g.score\_team1, g.score\_team2) AS won\_by\_score,

opp.team\_name AS opponent\_team

FROM Games g

JOIN Teams t

ON (g.team1\_id = t.team\_id AND g.score\_team1 > g.score\_team2)

OR (g.team2\_id = t.team\_id AND g.score\_team2 > g.score\_team1)

JOIN Teams opp

ON ( (t.team\_id = g.team1\_id AND opp.team\_id = g.team2\_id)

OR (t.team\_id = g.team2\_id AND opp.team\_id = g.team1\_id) )

ORDER BY g.game\_id ASC;

**-- 15. Determine the average number of rebounds per player for each team and list the teams in descending order of average rebounds.**

SELECT

t.team\_name,

AVG(ps.rebounds) AS avg\_team\_rebounds

FROM

PlayerStats AS ps

JOIN

Players AS p ON ps.player\_id = p.player\_id

JOIN

Teams AS t ON p.team\_id = t.team\_id

GROUP BY

t.team\_name

ORDER BY

avg\_team\_rebounds DESC;