NEXTLEAP - MONEYBALL CHALLENGE

Task.1 - Find the Top Performers

Query: Write a SQL query to list the top 5 players who have the highest number of goals scored. Include their name, team, goals scored, and position.

Expected Output: A table showing player names, their team names, goals scored, and their positions.

```sql
SELECT
p.name AS player_name,
t.team_name,
p.goals_scored,
p.position
FROM players p
JOIN teams t ON p.team_id = t.team_id
ORDER BY p.goals_scored DESC
LIMIT 5;
```

Solution Accepted

Your Answer

player_name	team_name	goals_scored	position
Erling Haaland	Manchester City	40	Forward
Robert Lewandowski	Juventus	35	Forward
Lionel Messi	FC Barcelona	30	Forward
Cristiano Ronaldo	Juventus	28	Forward
Kylian Mbappe	Paris Saint-Germain	27	Forward

Explanation: The query joins the **players** table with the **teams** table on **team_id** to get the team name for each player, then sorts by **goals_scored** in descending order and takes only the top 5 rows.

Task.2 - Best Value for Money Players

Query: Write a SQL query to identify the top 3 players who provide the best value for money.
Value for money is defined as the number of goals scored per unit of salary (goals/salary).
For readability, scale this ratio by multiplying by 1,000,000, and display the result to one decimal places based on precision.
Show player name, team name, goals scored, salary, and the calculated value-for-money ratio.
Sort the output in descending order of value_for_money_ratio.

Expected Output: A table with player names, their team names, goals, salaries, and their value-for-money ratio, sorted in descending order.

```sql
SELECT
p.name AS player_name,
t.team_name,
p.goals_scored,
p.salary,
    CAST ((p.goals_scored*10000000.0/p.salary)  AS INT)/10.0 AS value_for_money_ratio
FROM players p
JOIN teams t ON p.team_id = t.team_id
ORDER BY value_for_money_ratio DESC
LIMIT 3;
```

Solution Accepted

Your Answer

player_name	team_name	goals_scored	salary	value_for_money_ratio
Erling Haaland	Manchester City	40	30000000	1.3
Mohamed Salah	Paris Saint-Germain	25	25000000	1.0
Kylian Mbappe	Paris Saint-Germain	27	30000000	0.9

Explanation: The query joins **players** and **teams** tables to get team names. It calculates a **value-for-money ratio** (goals per salary × 1,000,000) using a CAST trick to simulate rounding to 1 decimal. Results are sorted by this ratio in descending order, and only the top 3 are returned.

Task.3 - Team Performance Summary

Query: Write a SQL query to generate a summary report of each team's performance. The report should show the team name, number of wins, losses, draws, total goals scored by all its players, and the team's league position.

Expected Output: A table listing team names, their number of wins, losses, draws, total goals, and league position.

```sql
SELECT
t.team_name,
t.wins,
t.losses,
t.draws,
CAST (TOTAL (p.goals_scored) AS INT) total_goals,
t.league_position
FROM teams t
JOIN players p ON t.team_id = p.team_id
GROUP BY t.team_id
ORDER BY t.league_position;
```

Solution Accepted

Your Answer

team_name	wins	losses	draws	total_goals	league_position
Paris Saint-Germain	22	2	3	62	1
Manchester City	21	4	1	65	2
FC Barcelona	20	3	2	33	3
Juventus	18	5	1	63	4

Explanation: This query generates a team performance summary by joining the **teams** and **players** tables on **team_id**, by **wins, losses, draws**. It aggregates total goals scored across all players per team using cast to **INT** for clean output. Results are grouped by team and ordered by league position in ascending order to rank from top to bottom of the league table.

Task.4 - Player Transfer Analysis

Query: Write a SQL query to find all players who have been transferred to a different team. Include player name, original team name, new team name, and the transfer fee.

Expected Output: A table showing player names, their original team, the team they transferred to, and the transfer fee.

```sql
SELECT
p.name AS player_name,
t1. team_name AS original_team,
t2. team_name AS new_team,
tr. transfer_fee
FROM transfers tr
JOIN players p ON tr. player_id = p.player_id
JOIN teams t1 ON tr.from_team_id = t1.team_id
JOIN teams t2 ON tr.to_team_id = t2.team_id
ORDER BY p.name;
```

Solution Accepted

Your Answer

player_name	original_team	new_team	transfer_fee
Kevin De Bruyne	FC Barcelona	Paris Saint-Germain	80000000
Sergio Ramos	Paris Saint-Germain	FC Barcelona	90000000
Virgil van Dijk	Juventus	Manchester City	70000000

Explanation: This query Joins the **transfers** table with **players** to get names, then joins **teams** twice — once as **T1** for the original team **(from_team_id)** and once as **T2** for the new team **(to_team_id)**. Results are sorted alphabetically by player name.

Task.5 - Identify Potential Underperformers

Query: Write a SQL query to identify players who have played more than 10 matches but have scored fewer than 4 goals. Include their name, team, matches played, and goals scored.

Expected Output: A table with player names, their team, number of matches played, and their goals scored.

```sql
SELECT
  p.name AS player_name,
  t. team_name,
  p. matches_played,
  p. goals_scored
  FROM players p
  JOIN teams t ON p.team_id = t.team_id
  WHERE p.matches_played > 10 AND p.goals_scored < 4;
```

Solution Accepted

Your Answer

player_name	team_name	matches_played	goals_scored
Sergio Ramos	FC Barcelona	21	3

Explanation: This query Joins **players** with **teams** for team names, then filters using **WHERE** to find players with more than 10 matches but fewer than 4 goals — identifying potential underperformers.

Task.6 -Team Budget Efficiency

Query: Write a SQL query to find the top 3 teams that have achieved the highest number of wins with the lowest budget. The "win-to-budget ratio" is calculated as the number of wins divided by the budget, scaled by a factor of 1,000,000, and rounded to two decimal places. Show the team name, number of wins, and their budget, sorted by the best win-to-budget ratio.

Expected Output: A table with team names, number of wins, and their budget, sorted by the best win-to-budget ratio

```sql
SELECT
t.team_name,
t.wins,
t.budget,
CAST (t.wins*100000000.0/t.budget + 0.5 AS INT)/ 100.0 AS win_to_budget_ratio
FROM teams t
ORDER BY win_to_budget_ratio DESC
LIMIT 3;
```

Solution Accepted

Your Answer

team_name	wins	budget	win_to_budget_ratio
Juventus	18	500000000	0.04
FC Barcelona	20	600000000	0.03
Paris Saint-Germain	22	700000000	0.03

Explanation: This query calculates **wins** per unit budget scaled by 1,000,000. It simulates rounding to 2 decimal places by multiplying by an extra 100, adding 0.5 (to round up when needed), casting to **INT**, then dividing by 100.0. Results are sorted by the ratio descending and limited to top 3.