SET A

Q1.

Marking scheme- this question is evaluated using binary marking scheme (0/1 Mark)

```
SELECT p.Name, t.Name AS TeamName, SUM(pr.RunsScored) AS TotalRuns
FROM Players p
JOIN Teams t ON p.TeamID = t.TeamID
JOIN Performance pr ON p.PlayerID = pr.PlayerID
GROUP BY p.PlayerID, p.Name, t.Name
ORDER BY TotalRuns DESC
LIMIT 10;
```

OR

```
SELECT p.Name, t.Name AS TeamName, r.TotalRuns
FROM (
SELECT PlayerID, SUM(RunsScored) AS TotalRuns
FROM Performance
GROUP BY PlayerID
ORDER BY TotalRuns DESC
LIMIT 10
) r
JOIN Players p ON p.PlayerID = r.PlayerID
JOIN Teams t ON p.TeamID = t.TeamID;
```

Q2.

Marking scheme- this question is evaluated using binary marking scheme (0/1 Mark)

```
SELECT t.Name, COUNT(*) AS Wins
```

```
FROM Matches m

JOIN Teams t ON m.WinnerTeamID = t.TeamID

JOIN Venues v ON m.VenueID = v.VenueID

WHERE v.City = &#39;City 55&#39;

GROUP BY t.TeamID, t.Name;
```

No. of records= 5

OR

```
SELECT t.Name, COUNT(*) AS Wins

FROM Matches m

JOIN Teams t ON m.WinnerTeamID = t.TeamID

WHERE VenueID IN (SELECT VenueID FROM Venues WHERE City = &#39;City 55&#39;)

GROUP BY t.TeamID;
```

Q3.

Marking scheme- this question is evaluated using binary marking scheme (0/1 Mark)

```
SELECT p.Name, pr.MatchID, pr.WicketsTaken

FROM Players p

JOIN Performance pr ON p.PlayerID = pr.PlayerID

WHERE pr.WicketsTaken &gt; 4 AND p.Role LIKE &#39;%bowler%&#39;;
```

OR

```
SELECT p.Name, pr.MatchID, pr.WicketsTaken

FROM Players p

JOIN Performance pr ON p.PlayerID = pr.PlayerID

JOIN matches mr ON mr.MatchID = pr.MatchID

WHERE pr.WicketsTaken &gt; 4 AND p.Role LIKE &#39;%bowler%&#39;;
```

OR

```
SELECT p.Name, pr.MatchID, pr.WicketsTaken
FROM (
SELECT PlayerID, Name
FROM Players
WHERE Role LIKE &#39;%bowler%&#39;
) p
JOIN Performance pr ON p.PlayerID = pr.PlayerID
WHERE pr.WicketsTaken &gt; 4;
```

Q4.

Marking scheme- this question is evaluated using binary marking scheme (0/1 Mark)

```
SELECT v.Name, v.City, COUNT(*) AS NumMatches
FROM Venues v
JOIN Matches m ON v.VenueID = m.VenueID
GROUP BY v.VenueID
ORDER BY NumMatches DESC
LIMIT 1;
```

OR

```
SELECT v.Name, v.City, COUNT(*) AS NumMatches
FROM Venues v
JOIN Matches m ON v.VenueID = m.VenueID
GROUP BY v.VenueID
HAVING NumMatches = (
SELECT MAX(NumMatches)

FROM (
SELECT VenueID, COUNT(*) AS NumMatches
FROM Matches
GROUP BY VenueID
) subquery
);
```

Q5. Marking scheme- this question is evaluated using binary marking scheme (0/1 Mark)

```
SELECT p.Name, SUM(pr.RunsScored) AS TotalRuns, SUM(pr.WicketsTaken) AS TotalWickets
FROM Players p
JOIN Performance pr ON p.PlayerID = pr.PlayerID
GROUP BY p.PlayerID
HAVING TotalRuns &gt; 100 AND TotalWickets &gt; 10;
```

OR

```
SELECT p.Name, r.TotalRuns, w.TotalWickets
FROM Players p
JOIN (
SELECT PlayerID, SUM(RunsScored) AS TotalRuns
FROM Performance
GROUP BY PlayerID
HAVING TotalRuns &gt; 100
) r ON p.PlayerID = r.PlayerID
JOIN (
SELECT PlayerID, SUM(WicketsTaken) AS TotalWickets
FROM Performance

GROUP BY PlayerID
HAVING TotalWickets &gt; 10
) w ON p.PlayerID = w.PlayerID;
```

Q6. Marking scheme- this question is evaluated using partial marking scheme (0/1/2 Marks)

Partial marks are awarded if the query is logically correct.

```
SELECT TeamID, PlayerID, Name, TotalRuns, TotalWickets,
RANK() OVER (PARTITION BY TeamID ORDER BY TotalRuns DESC, TotalWickets DESC) AS R
FROM (
SELECT t.TeamID, p.PlayerID, p.Name, SUM(pr.RunsScored) AS TotalRuns, SUM(pr.WicketsTaken)
AS TotalWickets
FROM Teams t
JOIN Players p ON t.TeamID = p.TeamID
JOIN Performance pr ON p.PlayerID = pr.PlayerID
GROUP BY t.TeamID, p.PlayerID, p.Name
) sub;
```

OR

```
SELECT
teamID,
playerID,
Name,
TotalRuns,
TotalWickets,
RANK() OVER (PARTITION BY teamID ORDER BY TotalRuns DESC, TotalWickets DESC) AS
PerformanceRank
FROM (
SELECT
p.teamID,
p.playerID,

p.Name,
SUM(pr.RunsScored) AS TotalRuns,
SUM(pr.WicketsTaken) AS TotalWickets
FROM
Players p
JOIN
Performance pr ON p.playerID = pr.playerID
GROUP BY
p.teamID, p.playerID, p.Name
) AS PlayerAggregates
ORDER BY
teamID, PerformanceRank;
```

Q7. Marking scheme- this question is evaluated using binary marking scheme (0/1 Mark)

```
SELECT p.Name, COUNT(*) AS Matches, AVG(pr.RunsScored) AS AverageScore
FROM Players p
JOIN Performance pr ON p.PlayerID = pr.PlayerID
WHERE pr.RunsScored &gt;= 30
GROUP BY p.PlayerID
HAVING COUNT(*) &gt; 5;
```

OR

```
SELECT p.Name, r.Matches, r.AverageScore
FROM Players p
JOIN (
SELECT PlayerID, COUNT(*) AS Matches, AVG(RunsScored) AS AverageScore

FROM Performance
WHERE RunsScored &gt;= 30
GROUP BY PlayerID
HAVING Matches &gt; 5
) r ON p.PlayerID = r.PlayerID;
```

Q8.

Marking scheme- this question is evaluated using partial marking scheme (0/1/2/3 Marks)

Partial marks are awarded if the query is logically correct.

```
DELIMITER //

CREATE FUNCTION GetBattingAverage1(player_id INT)
RETURNS FLOAT
READS SQL DATA
BEGIN
   DECLARE runs INT DEFAULT 0;
   DECLARE outs INT DEFAULT 0;
   SELECT SUM(RunsScored), COUNT(*) INTO runs, outs
   FROM Performance
   WHERE PlayerID = player_id AND RunsScored > 0;
   RETURN IF(outs = 0, NULL, runs / outs);
END;

DELIMITER //;
```

OR

```
drop function GetBattingAverage;
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION
`GetBattingAverage`(playerid INT) RETURNS decimal(10,2)
 DETERMINISTIC
BEGIN
 DECLARE avg_value decimal(10,2);

 set avg_value=
 (select (sum(pe.runsscored)/count(pe.matchid))
 from performance pe,players p
 where p.playerid=pe.playerid
 group by p.playerid
 having p.playerid=playerid);
 RETURN avg_value;
 END$$
DELIMITER ;
select GetBattingAverage(2);
```

OR

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION
`GetBattingAveragem1`(player_id INT) RETURNS FLOAT
deterministic
reads sql data
BEGIN
		declare total_runs INT;
    declare number_of_matches INT;
		declare avgBat FLOAT;
```

```
   SET total_runs = (
                        SELECT SUM(RunsScored)
                        FROM performance
       WHERE performance.PlayerID = player_id
       );
             SET number_of_matches = (
                        SELECT COUNT(MatchID)
       FROM performance
       WHERE performance.PlayerID = player_id
   );
   SET avgBat = total_runs/number_of_matches;
   return avgBat;
end$$
DELIMITER ;

SELECT players.Name, players.PlayerID, GetBattingAveragem1(players.PlayerID)
FROM players
WHERE PlayerID = 2;
```

Q9.

Marking scheme- this question is evaluated using partial marking scheme (0/1/2/3 Marks)

Partial marks are awarded if the query is logically correct.

```
SELECT
MatchID,
p.Name,
pr.RunsScored,
pr.WicketsTaken,
pr.CatchesTaken,
RANK() OVER (PARTITION BY MatchID ORDER BY (pr.RunsScored + pr.WicketsTaken * 20 +
pr.CatchesTaken * 10) DESC) AS PerformanceRank
FROM
Players p
JOIN
Performance pr ON p.PlayerID = pr.PlayerID
ORDER BY
PerformanceRank
LIMIT 1;
```

OR

```
SELECT pr.MatchID, p.Name, pr.RunsScored, pr.WicketsTaken, pr.CatchesTaken, PerformanceRank
FROM Players p
JOIN (
SELECT MatchID, PlayerID, RunsScored, WicketsTaken, CatchesTaken,
RANK() OVER (partition by matchid ORDER BY (RunsScored + WicketsTaken*20 +
CatchesTaken*10) DESC) AS PerformanceRank
FROM Performance
) pr ON p.PlayerID = pr.PlayerID limit 1;
```

Q10.

Marking scheme- this question is evaluated using partial marking scheme (0/1/2 Marks)

Partial marks are awarded if the query is logically correct.

```
SELECT p.Name, t.Name AS TeamName, t.City, (SUM(pr.RunsScored) / COUNT(pr.PlayerID)) AS
BattingAvg
FROM Players p
JOIN Teams t ON p.TeamID = t.TeamID
JOIN Performance pr ON p.PlayerID = pr.PlayerID

GROUP BY p.PlayerID, p.Name, t.Name, t.City

ORDER BY BattingAvg DESC

LIMIT 5;
```

OR

```
SELECT p.Name, t.Name AS TeamName, t.City, r.BattingAvg

FROM (

    SELECT PlayerID, SUM(RunsScored) / COUNT(*) AS BattingAvg

    FROM Performance

    GROUP BY PlayerID

    ORDER BY BattingAvg DESC

    LIMIT 5

) r

JOIN Players p ON p.PlayerID = r.PlayerID

JOIN Teams t ON p.TeamID = t.TeamID;
```

OR

```
select players.name, teams.name, teams.city, GetBattingAverage(players.playerID) as "BattingAverage"

from players

join

teams on players.TeamID = teams.TeamID

order by BattingAverage desc limit 5;
```

OR

```
select p.name, t.name, t.city, (sum(pe.runsscored) over (partition by p.playerid)/count(pe.matchid) over
(partition by p.playerid)) as avg_batt
```

```
from players p, teams t, performance pe

where p.teamid=t.teamid and p.playerid=pe.playerid

order by avg_batt desc

limit 5;
```

Q11.

Marking scheme- this question is evaluated using partial marking scheme (0/1/2/3 Marks)

Partial marks are awarded if the query is logically correct.

```
SELECT p.Name, t.Name AS TeamName, SUM(pr.RunsScored) AS TotalRuns, SUM(pr.CatchesTaken) AS
TotalCatches, SUM(pr.StumpsMade) AS TotalStumps

FROM Players p

JOIN Teams t ON p.TeamID = t.TeamID

JOIN Performance pr ON p.PlayerID = pr.PlayerID

GROUP BY p.PlayerID, p.Name, t.Name;
```

OR

```
select players.Name,teams.Name as team_name, sum(performance.RunsScored) as total_runs,
sum(performance.CatchesTaken)

as total_catches,sum(performance.StumpsMade) as total_stumps

from players,teams,performance

where players.TeamID=teams.TeamID and players.PlayerID=performance.PlayerID

group by players.PlayerID;
```

OR

```
select pl.name as name, t.name as team_name, sum(per.runsscored) as total_runs_scored,
sum(per.wicketstaken) as total_wickets, sum(per.catchestaken) as total_catches, sum(per.stumpsmade)
as total_stumps

from players pl

inner join teams t

on t.teamid = pl.teamid

inner join performance per

ON per.playerid = pl.playerid

group by per.playerid;
```

OR

```
SELECT

    p.Name AS PlayerName,

    t.Name AS TeamName,

    SUM(pf.RunsScored) AS TotalRuns,

    SUM(pf.CatchesTaken) AS TotalCatches,

    SUM(pf.StumpsMade) AS TotalStumps

FROM

    Players p

JOIN

    Teams t ON p.TeamID = t.TeamID

JOIN

    Performance pf ON p.PlayerID = pf.PlayerID

GROUP BY

    p.PlayerID, p.Name, t.Name;
```

Q12.

Marking scheme- this question is evaluated using partial marking scheme (0/1/2/3 Marks)

Partial marks are awarded if the query is logically correct.

```
CREATE VIEW HighScoringVenues AS
SELECT v.VenueID, v.Name, v.City, AVG(s.TotalScore) AS AverageTotalScore
FROM Venues v
JOIN (
    SELECT m.VenueID, s.MatchID, SUM(s.Runs) AS TotalScore
    FROM Scores s
    JOIN Matches m ON s.MatchID = m.MatchID
    GROUP BY s.MatchID
) s ON v.VenueID = s.VenueID
GROUP BY v.VenueID
HAVING AverageTotalScore > 150 AND COUNT(s.MatchID) > 2;


-- Query using the view
SELECT Name, City, AverageTotalScore
FROM HighScoringVenues;
```

OR

```
CREATE VIEW HighScoringVenuesasa1 AS
SELECT
    v.VenueID,
    v.Name AS VenueName,
    v.City,
    AVG(total.Runs) AS AverageTotalScore,
    COUNT(*) AS MatchesPlayed
FROM
    Venues v
JOIN
    Matches m ON v.VenueID = m.VenueID
JOIN
    (
        SELECT
            MatchID,
            SUM(Runs) AS Runs -- Total runs in each match considering all innings
        FROM
            Scores
        GROUP BY
            MatchID
    ) total ON total.MatchID = m.MatchID
GROUP BY
    v.VenueID, v.Name, v.City
HAVING
    AVG(total.Runs) > 150 AND COUNT(*) > 2;
    select * from HighScoringVenuesasa1;
```

*****The End*****