**Performance Table**

**Table Recap**

CREATE TABLE Performance (

Name VARCHAR(50),

Score INT,

IsPassed BOOLEAN,

Percentage DECIMAL(5,2)

);

INSERT INTO Performance (Name, Score, IsPassed, Percentage) VALUES

('Aarav Mehta', 85, TRUE, 85.00),

('Isha Kapoor', 72, TRUE, 72.00),

('Rohan Singh', 48, FALSE, 48.00),

('Neha Verma', 91, TRUE, 91.00),

('Kunal Joshi', 66, TRUE, 66.00),

('Priya Sharma', 59, FALSE, 59.00),

('Aditya Rao', 77, TRUE, 77.00),

('Sneha Das', 34, FALSE, 34.00),

('Manav Patel', 88, TRUE, 88.00),

('Tanya Jain', 95, TRUE, 95.00);

**🔹 Part A: Aggregations & Filters (WHERE + HAVING)**

**Q1.** Find the highest, lowest, and average score of students.

SELECT MAX(Score) AS Highest, MIN(Score) AS Lowest, AVG(Score) AS Avg\_Score

FROM Performance;

**Q2.** Count how many students passed and failed.

SELECT IsPassed, COUNT(\*) AS Total

FROM Performance

GROUP BY IsPassed;

**Q3.** Show the average score of only passed students (WHERE).

SELECT AVG(Score) AS Avg\_Passed\_Score

FROM Performance

WHERE IsPassed = TRUE;

**Q4.** Find total students having percentage above 70 (HAVING).

SELECT IsPassed, COUNT(\*) AS Students

FROM Performance

GROUP BY IsPassed

HAVING AVG(Percentage) > 70;

**Q5.** Display students who scored above the overall class average.

SELECT Name, Score

FROM Performance

WHERE Score > (SELECT AVG(Score) FROM Performance);

**🔹 Part B: Ranking & Numbering Functions**

**Q6.** Rank students by score (highest first).

SELECT Name, Score, RANK() OVER (ORDER BY Score DESC) AS Rank\_Pos

FROM Performance;

**Q7.** Assign a unique row number to each student.

SELECT Name, Score, ROW\_NUMBER() OVER (ORDER BY Score DESC) AS RowNum

FROM Performance;

**Q8.** Assign dense rank (no gaps in rank).

SELECT Name, Score, DENSE\_RANK() OVER (ORDER BY Score DESC) AS DenseRank\_Pos

FROM Performance;

**Q9.** Divide students into 4 quartiles based on score.

SELECT Name, Score, NTILE(4) OVER (ORDER BY Score DESC) AS Quartile

FROM Performance;

**Q10.** Divide students into 2 groups: top 50% and bottom 50%.

SELECT Name, Score, NTILE(2) OVER (ORDER BY Score DESC) AS Half\_Group

FROM Performance;

**🔹 Part C: Window Aggregate Functions**

**Q11.** Show each student’s score and overall class average.

SELECT Name, Score, AVG(Score) OVER () AS Class\_Avg

FROM Performance;

Select ROUND (score,2) as rounded from Performance

**Q12.** Calculate running total of scores (ordered).

SELECT Name, Score, SUM(Score) OVER (ORDER BY Score) AS Running\_Total

FROM Performance;

**Q13.** Show each student’s score and max score in class.

SELECT Name, Score, MAX(Score) OVER () AS Max\_Score

FROM Performance;

**Q14.** Show each student’s score and average score among pass/fail group.

SELECT Name, IsPassed, Score,

AVG(Score) OVER (PARTITION BY IsPassed) AS Group\_Avg

FROM Performance;

**Q15.** Calculate cumulative average score ordered by marks.

SELECT Name, Score, AVG(Score) OVER (ORDER BY Score ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS Running\_Avg

FROM Performance;

**🔹 Part D: LEAD & LAG Functions**

**Q16.** Show each student’s score and the next student’s score.

SELECT Name, Score,

LEAD(Score) OVER (ORDER BY Score DESC) AS Next\_Score

FROM Performance;

**Q17.** Show each student’s score and the previous student’s score.

SELECT Name, Score,

LAG(Score) OVER (ORDER BY Score DESC) AS Prev\_Score

FROM Performance;

**Q18.** Compare each student’s score difference with next student.

SELECT Name, Score,

LEAD(Score) OVER (ORDER BY Score DESC) - Score AS Diff\_With\_Next

FROM Performance;

**Q19.** Show pass/fail students and their rank inside each group.

SELECT Name, IsPassed, Score,

RANK() OVER (PARTITION BY IsPassed ORDER BY Score DESC) AS Rank\_In\_Group

FROM Performance;

**Q20.** Show students with best & worst score in each group (pass/fail).

SELECT Name, IsPassed, Score,

FIRST\_VALUE(Name) OVER (PARTITION BY IsPassed ORDER BY Score DESC) AS Top\_Student,

LAST\_VALUE(Name) OVER (PARTITION BY IsPassed ORDER BY Score ASC RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS Lowest\_Student

FROM Performance;

**🔹 Part E: Lab Work (Hands-on Practice)**

**Q21.** Write a query to find the **top 3 scorers** using LIMIT or RANK().

**Q22.** Write a query to calculate the **percentage difference between each student and the class average**.

**Q23.** Write a query to find the **second-highest score** without using MAX().

**Q24.** Use NTILE(5) to divide students into **percentile groups** and display them.

**Q25.** Create a query that shows each student along with **cumulative pass count** (running total of students who passed).

**🔹 Lab Work Queries with Answers**

**Q21. Write a query to find the top 3 scorers using LIMIT or RANK().**

✅ Using LIMIT:

SELECT Name, Score

FROM Performance

ORDER BY Score DESC

LIMIT 3;

✅ Using RANK():

SELECT Name, Score

FROM (

SELECT Name, Score,

RANK() OVER (ORDER BY Score DESC) AS rnk

FROM Performance

) t

WHERE rnk <= 3;

👉 **Explanation:**

* LIMIT → simply takes top 3 rows.
* RANK() → useful if there are ties (e.g., multiple students at 3rd place).

**Q22. Write a query to calculate the percentage difference between each student and the class average.**

SELECT

Name, Score,

AVG(Score) OVER () AS Class\_Avg,

ROUND(((Score - AVG(Score) OVER ()) / AVG(Score) OVER ()) \* 100, 2) AS Diff\_Percent

FROM Performance;

👉 **Explanation:**

* AVG() OVER () → gives overall class average.
* Difference % = (Score - ClassAvg) / ClassAvg \* 100.

**Q23. Write a query to find the second-highest score without using MAX().**

✅ Using DISTINCT + ORDER BY:

SELECT DISTINCT Score

FROM Performance

ORDER BY Score DESC

OFFSET 1 ROW FETCH NEXT 1 ROW ONLY;

✅ Or with RANK():

SELECT Name, Score

FROM (

SELECT Name, Score, RANK() OVER (ORDER BY Score DESC) AS rnk

FROM Performance

) t

WHERE rnk = 2;

👉 **Explanation:**

* OFFSET skips the top row.
* RANK() picks students with rank = 2.

**Q24. Use NTILE(5) to divide students into percentile groups and display them.**

SELECT

Name, Score,

NTILE(5) OVER (ORDER BY Score DESC) AS Percentile\_Group

FROM Performance;

👉 **Explanation:**

* NTILE(5) splits ordered data into 5 buckets → simulating **percentile ranges (20% each)**.

**Q25. Create a query that shows each student along with cumulative pass count (running total of students who passed).**

SELECT

Name, IsPassed,

SUM(CASE WHEN IsPassed = TRUE THEN 1 ELSE 0 END)

OVER (ORDER BY Score) AS Cumulative\_Pass\_Count

FROM Performance

ORDER BY Score;

👉 **Explanation:**

* CASE checks if student passed (adds 1).
* SUM() OVER (ORDER BY Score) → cumulative total across scores.

**🔹 10 More SQL Questions (Filter-focused)**

**Q26.** Show all students who scored **above 80**.

SELECT Name, Score

FROM Performance

WHERE Score > 80;

**Q27.** Show all students who **failed (IsPassed = FALSE)**.

SELECT Name, Score, IsPassed

FROM Performance

WHERE IsPassed = FALSE;

**Q28.** Display students whose score is **between 60 and 90**.

SELECT Name, Score

FROM Performance

WHERE Score BETWEEN 60 AND 90;

**Q29.** Show students whose name starts with **‘A’**.

SELECT Name, Score

FROM Performance

WHERE Name LIKE 'A%';

**Q30.** Show students whose name ends with **‘a’**.

SELECT Name, Score

FROM Performance

WHERE Name LIKE '%a';

**Q31.** Find students with names containing **‘sh’**.

SELECT Name, Score

FROM Performance

WHERE Name LIKE '%sh%';

**Q32.** Show students whose score is **NOT between 50 and 70**.

SELECT Name, Score

FROM Performance

WHERE Score NOT BETWEEN 50 AND 70;

**Q33.** Show students whose names are in the list **(‘Aarav Mehta’, ‘Neha Verma’, ‘Tanya Jain’)**.

SELECT Name, Score

FROM Performance

WHERE Name IN ('Aarav Mehta', 'Neha Verma', 'Tanya Jain');

**Q34.** Show students whose names are **not in the above list**.

SELECT Name, Score

FROM Performance

WHERE Name NOT IN ('Aarav Mehta', 'Neha Verma', 'Tanya Jain');

**Q35.** Show all students who **passed** and have a score greater than **85**.

SELECT Name, Score, IsPassed

FROM Performance

WHERE IsPassed = TRUE AND Score > 85;