1. Which of the following correctly defines a window function in MySQL?

A. A function that aggregates rows into a single result for each group
B. A function that performs calculations across a set of rows related to the current row
C. A function that creates temporary tables for joins
D. A function that partitions data into multiple databases
✅ **ANSWER:** B
🧩 **Explanation:**
A window function performs a calculation across a related set of rows without collapsing them into a single result (unlike GROUP BY). Examples include `RANK()`, `ROW_NUMBER()`, and `SUM() OVER(...)`.

---

2. Which clause is mandatory when using a window function like `RANK()`?

A. GROUP BY
B. ORDER BY
C. OVER()
D. LIMIT
✅ **ANSWER:** C
🧩 **Explanation:**
Every window function requires the `OVER()` clause to define the window scope. Without it, MySQL cannot apply the function per row context. The `OVER()` can be empty or include `PARTITION BY` / `ORDER BY`.

---

3. What will the following query return?

```
SELECT emp_name, salary, RANK() OVER (ORDER BY salary DESC) AS position FROM
employees;
```

A. A unique rank even for equal salaries
B. The same rank for equal salaries, with gaps in the sequence
C. A running total of salaries
D. The average salary per department
✅ **ANSWER:** B
🧩 **Explanation:**
`RANK()` assigns equal rank for equal values but skips subsequent ranks. If two employees share salary rank 1, the next rank will be 3.

---

4. In MySQL, what is the difference between `RANK()` and `DENSE_RANK()`?

A. `DENSE_RANK()` skips ranks for ties, while `RANK()` doesn't
B. `RANK()` skips ranks for ties, while `DENSE_RANK()` doesn't
C. Both behave identically
D. `DENSE_RANK()` can't be used with PARTITION BY
✅ **ANSWER:** B
🧩 **Explanation:**

`RANK()` produces gaps after ties (1, 2, 2, 4), while `DENSE_RANK()` does not (1, 2, 2, 3). Both can use `PARTITION BY`.

---

5. What is the purpose of the `PARTITION BY` clause in a window function?

A. To filter rows using a condition
B. To divide result sets into groups before applying the window function
C. To limit the number of returned rows
D. To sort results in ascending order
✅ **ANSWER:** B
🧩 **Explanation:**
`PARTITION BY` divides data into subsets (windows), and the function operates separately within each partition — for example, calculating ranks within each department.

---

6. What will the following query compute?

```
SELECT emp_id, department, salary, SUM(salary) OVER (PARTITION BY department) AS
dept_total FROM employees;
```

A. The total salary of all employees
B. The total salary per department
C. The average salary across all employees
D. The cumulative salary ordered by department
✅ **ANSWER:** B
🧩 **Explanation:**
The query sums salaries per department and displays the same department total for each employee, without grouping or collapsing rows.

---

7. What will this CASE statement output when salary = 80000?

```
CASE WHEN salary > 100000 THEN 'High' WHEN salary >= 70000 THEN 'Medium' ELSE 'Low'
END
```

A. High
B. Medium
C. Low
D. NULL
✅ **ANSWER:** B
🧩 **Explanation:**
The first true condition (`salary >= 70000`) matches, so `'Medium'` is returned. CASE executes sequentially and stops after the first match.

---

8. Which of the following is TRUE about CASE statements in MySQL?

A. CASE must always end with an ELSE clause
B. CASE can be used both in SELECT and ORDER BY clauses
C. CASE can only compare numeric columns
D. CASE cannot be nested
✅ **ANSWER:** B
🧩 **Explanation:**
CASE expressions are flexible — they can be used in SELECT, ORDER BY, GROUP BY, and WHERE. The ELSE clause is optional, and CASE can handle text, numbers, or dates.

---

9. What is the difference between `ROW_NUMBER()` and `RANK()` functions in MySQL?

A. `ROW_NUMBER()` gives unique sequential numbers, `RANK()` gives same numbers for ties
B. `RANK()` gives unique numbers, `ROW_NUMBER()` gives same numbers for ties
C. Both are identical
D. `ROW_NUMBER()` cannot be used with ORDER BY
✅ **ANSWER:** A
🧩 **Explanation:**
`ROW_NUMBER()` assigns unique, consecutive numbers regardless of ties. `RANK()` gives equal rank to ties and skips subsequent numbers.

---

10. Consider this query:

```
SELECT emp_name, department, salary, CASE WHEN salary > AVG(salary) OVER (PARTITION BY department) THEN 'Above Avg' ELSE 'Below Avg' END AS performance FROM employees;
```

What does this query compute?
A. Employees grouped by salary brackets
B. Compares each employee's salary with department average using a window function
C. Finds total salary per department
D. Calculates cumulative salary rank
✅ **ANSWER:** B
🧩 **Explanation:**
The windowed `AVG()` calculates the department's average per row. The CASE compares each employee's salary to that average and labels as "Above Avg" or "Below Avg".