# Top SQL Functions

**Must Know for Interviews**

*(with example queries)*

**CloudyData**

swipe

# Advanced SQL Functions with Examples

## 🔄 Window Functions

1. **ROW_NUMBER()**
   Assigns a unique sequential number to rows within a partition.
   - *Example:*
     `SELECT name, ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_num FROM employees`
2. **RANK()**
   Provides the rank of rows within a partition, with gaps for ties.
   - *Example:*
     `SELECT name, RANK() OVER (ORDER BY marks DESC) AS rank FROM students`
3. **DENSE_RANK()**
   Similar to RANK(), but without gaps in ranking values.
   - *Example:*
     `SELECT name, DENSE_RANK() OVER (ORDER BY salary DESC) AS rank FROM employees`
4. **NTILE(n)**
   Divides rows into 'n' approximately equal groups.
   - *Example:*
     `SELECT name, NTILE(4) OVER (ORDER BY score DESC) AS quartile FROM students`
5. **LAG()**
   Accesses data from a previous row in the same result set.
   - *Example:*
     `SELECT name, salary, LAG(salary) OVER (ORDER BY salary) AS previous_salary FROM employees`

6. **LEAD()**
   Accesses data from the next row in the same result set.

   - *Example:*
     ```
     SELECT name, salary, LEAD(salary) OVER (ORDER BY salary) AS
     next_salary FROM employees
     ```

7. **FIRST_VALUE()**
   Returns the first value in an ordered set of values.

   - *Example:*
     ```
     SELECT name, FIRST_VALUE(salary) OVER (ORDER BY salary
     DESC) AS highest_salary FROM employees
     ```

8. **LAST_VALUE()**
   Returns the last value in an ordered set of values.

   - *Example:*
     ```
     SELECT name, LAST_VALUE(salary) OVER (ORDER BY salary DESC
     ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS
     lowest_salary FROM employees
     ```

9. **CUME_DIST()**
   Calculates the cumulative distribution of a value in a group.

   - *Example:*
     ```
     SELECT name, CUME_DIST() OVER (ORDER BY salary) AS cum_dist
     FROM employees
     ```

10. **PERCENT_RANK()**
    Calculates the relative rank of a row within a group.

    - *Example:*
      ```
      SELECT name, PERCENT_RANK() OVER (ORDER BY salary) AS
      percent_rank FROM employees
      ```

## 📊 Aggregate Functions

11. **SUM()**
    Calculates the total sum of a numeric column.
    - *Example:*
      ```
      SELECT SUM(salary) AS total_salary FROM employees
      ```
12. **AVG()**
    Computes the average value of a numeric column.
    - *Example:*
      ```
      SELECT AVG(age) AS average_age FROM users
      ```
13. **COUNT()**
    Counts the number of rows or non-NULL values.
    - *Example:*
      ```
      SELECT COUNT(*) AS total_employees FROM employees
      ```
14. **MIN() / MAX()**
    Retrieves the minimum or maximum value in a column.
    - *Example:*
      ```
      SELECT MIN(price) AS lowest_price, MAX(price) AS
      highest_price FROM products
      ```
15. **GROUP_CONCAT() / STRING_AGG()**
    Concatenates values from multiple rows into a single string.
    - *Example:*
      ```
      SELECT department, STRING_AGG(name, ', ') AS employee_names
      FROM employees GROUP BY department
      ```

## 🧠 String Functions

16. **CONCAT()**
    Combines two or more strings into one.
    - *Example:*
      `SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM users`

17. **SUBSTRING()**
    Extracts a portion of a string.
    - *Example:*
      `SELECT SUBSTRING(name, 1, 3) AS short_name FROM users`

18. **REPLACE()**
    Replaces occurrences of a substring within a string.
    - *Example:*
      `SELECT REPLACE(name, 'John', 'Jonathan') AS updated_name FROM users`

19. **TRIM() / LTRIM() / RTRIM()**
    Removes specified characters from the beginning and/or end of a string.
    - *Example:*
      `SELECT TRIM(' ' FROM name) AS trimmed_name FROM users`

20. **UPPER() / LOWER()**
    Converts strings to uppercase or lowercase.
    - *Example:*
      `SELECT UPPER(name) AS uppercase_name FROM users`

21. **LENGTH()**
    Returns the length of a string.
    - *Example:*
      `SELECT LENGTH(name) AS name_length FROM users`

22. **CHARINDEX() / INSTR()**
Finds the position of a substring within a string.

- *Example:*
```
SELECT CHARINDEX('a', name) AS position FROM users
```

---

## 📅 Date & Time Functions

23. **NOW() / CURRENT_TIMESTAMP**
Retrieves the current date and time.

- *Example:*
```
SELECT NOW() AS current_datetime
```

24. **DATEADD()**
Adds a specified time interval to a date.

- *Example:*
```
SELECT DATEADD(day, 7, order_date) AS delivery_date FROM
orders
```

25. **DATEDIFF()**
Calculates the difference between two dates.

- *Example:*
```
SELECT DATEDIFF(day, order_date, delivery_date) AS
days_between FROM orders
```

26. **DATEPART()**
Extracts a specific part of a date (e.g., year, month).

- *Example:*
```
SELECT DATEPART(year, hire_date) AS hire_year FROM
employees
```

27. **YEAR() / MONTH() / DAY()**
Retrieves the year, month, or day from a date.

- *Example:*
  `SELECT YEAR(birth_date) AS birth_year FROM users`

## 🔢 Mathematical Functions

28. **ROUND()**
    Rounds a numeric value to a specified number of decimal places.
    - *Example:*
      `SELECT ROUND(salary, 2) AS rounded_salary FROM employees`
29. **CEILING() / FLOOR()**
    Rounds a number up or down to the nearest integer.
    - *Example:*
      `SELECT CEILING(price) AS rounded_up_price FROM products`
30. **ABS()**
    Returns the absolute value of a number.
    - *Example:*
      `SELECT ABS(balance) AS absolute_balance FROM accounts`
31. **POWER()**
    Raises a number to the power of another number.
    - *Example:*
      `SELECT POWER(base, exponent) AS result FROM calculations`
32. **SQRT()**
    Calculates the square root of a number.
    - *Example:*
      `SELECT SQRT(area) AS side_length FROM squares`
33. **MOD() / %**
    Returns the remainder of a division operation.
    - *Example:*
      `SELECT MOD(score, 2) AS remainder FROM results`

## 🔍 Conditional & Null Handling Functions

34. **CASE WHEN**
    Implements conditional logic within queries.
    - *Example:*
      ```
      SELECT name, CASE WHEN score >= 90 THEN 'A' ELSE 'B' END AS grade FROM students
      ```

35. **COALESCE()**
    Returns the first non-NULL value in a list.
    - *Example:*
      ```
      SELECT COALESCE(middle_name, 'N/A') AS middle FROM users
      ```

36. **NULLIF()**
    Returns NULL if two expressions are equal.
    - *Example:*
      ```
      SELECT NULLIF(salary, bonus) AS difference FROM employees
      ```

37. **ISNULL()**
    Replaces NULL with a specified replacement value.
    - *Example:*
      ```
      SELECT ISNULL(phone, 'Not Provided') AS contact_number FROM users
      ```

## 🔄 Data Transformation Functions

38. **PIVOT / UNPIVOT**
    Rotates rows into columns and vice versa.

    *Example:*
    ```
    -- PIVOT example varies by SQL dialect
    ```

39. **CAST() / CONVERT()**
Converts data from one type to another.

*Example:*
```
SELECT CAST(price AS DECIMAL(10,2)) AS formatted_price FROM
products
```

---

## 📁 Common Table Expressions (CTEs)

40. **WITH Clause (CTE)**
Defines a temporary result set for use within a query.

- *Example:*
```
WITH recent_orders AS (SELECT * FROM orders WHERE
order_date > '2025-01-01') SELECT * FROM recent_orders
```

41. **Recursive CTEs**
Allows a CTE to reference itself for hierarchical data.

- *Example:*
```
WITH RECURSIVE employee_hierarchy AS (SELECT id, manager_id
FROM employees WHERE manager_id IS NULL UNION ALL SELECT
e.id, e.manager_id FROM employees e INNER JOIN
employee_hierarchy eh ON e.manager_id = eh.id) SELECT * FROM
employee_hierarchy
```

## 🧩 Advanced Query Techniques

42. **Subqueries**
A query nested within another SQL query.

- *Example:*
```
SELECT name FROM employees WHERE salary > (SELECT
AVG(salary) FROM employees)
```

43. **EXISTS / NOT EXISTS**
Tests for the existence of rows in a subquery.

- *Example:*
```
SELECT name FROM customers WHERE EXISTS (SELECT 1 FROM
orders WHERE customers.id = orders.customer_id)
```

44. **EXCEPT / INTERSECT**
Returns distinct rows from one query that are not in another (EXCEPT) or common to both (INTERSECT).

- *Example:*
```
SELECT name FROM employees EXCEPT SELECT name FROM retirees
```

## 🔐 System & Metadata Functions

45. **CURRENT_USER / SESSION_USER**
Returns the name of the current user.

- *Example:*
```
SELECT CURRENT_USER
```

46. **DB_NAME() / OBJECT_NAME()**
Retrieves the name of the current database or object.

- *Example:*
```
SELECT DB_NAME() AS database_name
```

47. **SYSTEM_USER**
Returns the login name for the current user.

- *Example:*
```
SELECT SYSTEM_USER
```

**Comment 'Function' to get pdf version of this post**