**# Important SQL Functions**

**• SUBSTRING Function :**

Syntax: SUBSTRING(string, position, length)

* The SUBSTRING() function returns a substring. It returns NULL if any arguments (string, position, and length) are NULL

**Eg:**

1. SELECT SUBSTRING('NISHANT', 1, 3) AS result;

*Result*

------

NIS

1. SELECT SUBSTRING('COMPUTER SCIENCE', 10, 7) AS result;

*Result*

------

SCIENCE

1. SELECT

email,

SUBSTRING(email, CHARINDEX('@', email) + 1, LEN(email)) AS domain

FROM students;

*Result*

------

| **email** | **domain** |
| --- | --- |
| [nishant@gmail.com](mailto:nishant@gmail.com) | gmail.com |
| [rajat@yahoo.com](mailto:rajat@yahoo.com) | yahoo.com |
| [user@iitm.ac.in](mailto:user@iitm.ac.in) | iitm.ac.in |

1. SELECT RIGHT('NISHANT', 3) AS result;

*Result*

------

ANT

5.) SELECT emp\_code, SUBSTRING(emp\_code, LEN(emp\_code) - 3, 4) AS last\_digits

FROM employees;

*Result*

------

| **emp\_code** | **last\_digits** |
| --- | --- |
| EMP12345 | 2345 |
| EMP56789 | 6789 |
| EMP99999 | 9999 |

**• CASE Function :**

Syntax:

CASE expression

WHEN when\_expression\_1 THEN result\_1

WHEN when\_expression\_2 THEN result\_2

WHEN when\_expression\_3 THEN result\_3

ELSE else\_result

END

**Eg:**

1. SELECT grade,

CASE grade

WHEN 'A' THEN 'Excellent'

WHEN 'B' THEN 'Good'

WHEN 'C' THEN 'Average'

ELSE 'Fail'

END AS remarks

FROM students;

*Result*

------

| **grade** | **remarks** |
| --- | --- |
| A | Excellent |
| B | Good |
| C | Average |
| F | Fail |
|  |  |

1. SELECT

salary,

CASE

WHEN salary >= 80000 THEN 'High'

WHEN salary BETWEEN 50000 AND 79999 THEN 'Medium'

ELSE 'Low'

END AS salary\_level

FROM employees;

*Result*

------

| **salary** | **salary\_level** |
| --- | --- |
| 90000 | High |
| 60000 | Medium |
| 30000 | Low |
|  |  |

1. SELECT

product,

price,

quantity,

CASE

WHEN quantity > 100 THEN price \* 0.9 -- 10% discount

WHEN quantity BETWEEN 50 AND 100 THEN price \* 0.95 -- 5% discount

ELSE price

END AS discounted\_price

FROM sales;

*Result*

------

| **product** | **price** | **quantity** | **discounted\_price** |
| --- | --- | --- | --- |
| Laptop | 60000 | 120 | 54000 |
| Mouse | 500 | 70 | 475 |
| Cable | 200 | 20 | 200 |
|  |  |  |  |

**• REPLACE Function :**

Syntax: REPLACE(string, search\_string, replacement\_string);

**Eg:**

1. SELECT REPLACE('We Will, We Will Rock You!', 'We', 'SQL') message;

*message*

------

SQL Will, SQL Will Rock You!

1. SELECT REPLACE('123-456-7890', '-', '') AS clean\_number;
2. SELECT

city\_name,

REPLACE(city\_name, '-', ' ') AS clean\_city

FROM cities;

*Result*

--------

| **city\_name** | **clean\_city** |
| --- | --- |
| New-Delhi | New Delhi |
| Surat-Old-Town | Surat Old Town |
| Ahmedabad-New | Ahmedabad New |

1. UPDATE students

SET address = REPLACE(address, 'Collage', 'College');

- Updates all rows where “Collage” appears.

**• CHARINDEX Function :**

Syntax: CHARINDEX(substring, string, [start\_position])

**Eg:**

1. SELECT CHARINDEX('shan', 'Nishant') AS position;

*Result*

--------

3

1. SELECT CHARINDEX('z', 'Nishant') AS position;

*Result*

--------

0

1. SELECT CHARINDEX('i', 'Nishant', 2) AS position;

*Result*

--------

0

- It starts searching from position 2, so it skips the first 'i'

1. SELECT

email,

SUBSTRING(email, 1, CHARINDEX('@', email) - 1) AS username

FROM students;

*Result*

--------

| **email** | **username** |
| --- | --- |
| [nishant@gmail.com](mailto:nishant@gmail.com) | nishant |
| [rajat@yahoo.com](mailto:rajat@yahoo.com) | rajat |

1. SELECT SUBSTRING('New Delhi', 1, CHARINDEX(' ', 'New Delhi') - 1) AS first\_word;

*Result*

--------

New

1. SELECT INSTR('NISHANT', 'A') AS position;

*Result*

--------

5

- It also works same as CHARINDEX Function But it will work on MySQL / Oracle / SQLite / PostgreSQL while CHARINDEX Function only work on SQL Server.

**• SUBSTRING Function :**

Syntax: CONCAT(string1, string2,..);

* The CONCAT function returns a string which is the combination of the input strings. It returns NULL if one of the arguments is NULL, also the result is NULL in SQL Server but ignored in MySQL

**Eg:**

1. SELECT CONCAT('Nishant', ' ', 'Kumar') AS full\_name;

*Result*

--------

Nishant Kumar

1. SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name

FROM students;

1. SELECT CONCAT('Order ID: ', order\_id, ', Amount: ₹', total) AS summary

FROM orders;

*Result*

--------

Order ID: 101, Amount: ₹1200

Order ID: 102, Amount: ₹500

1. SELECT CONCAT\_WS('-', '2025', '10', '11') AS date\_str;

*Result*

--------

2025-10-11

- CONCAT\_WS() = CONCAT With Separator, Very useful for joining columns with a specific separator like commas, slashes, etc.

1. SELECT CONCAT('Hello ', NULL, ' Nishant');

- In MySQL / PostgreSQL / Oracle → 'Hello Nishant' (NULL ignored)

- In SQL Server → NULL (because NULL makes the whole string NULL)

**• TRIM, LTRIM, and RTRIM Function :**

Syntax: TRIM([characters] FROM string)

**Eg:**

1. SELECT TRIM('#' FROM '###Hello###') AS result;

*Result*

--------

Hello

1. SELECT TRIM(' Nishant ') AS cleaned;

*Result*

--------

Nishant

1. SELECT RTRIM('Nishant ') AS result;

*Result*

--------

Nishant

1. SELECT LTRIM(RTRIM(' Nishant ')) AS cleaned;

*Result*

--------

Nishant

1. SELECT REPLACE(TRIM(name), ' ', ' ') AS fixed\_name

FROM students;

- Removes leading/trailing spaces, then replaces double spaces with single.

**• ROUND Function :**

Syntax: ROUND(num, d)

**Eg:**

1. ROUND(12.3456, 2) → 12.35

**• CEIL / CEILING Function :**

Syntax: CEIL(num)

**Eg:**

1. CEIL(4.2) → 5

**• FLOOR Function :**

Syntax: FLOOR(num)

**Eg:**

1. FLOOR(4.9) → 4

**• ABS Function :**

Syntax: ABS(num)

**Eg:**

1. ABS(-5) → 5

**• POWER Function :**

Syntax: POWER(a,b)

**Eg:**

1. POWER(2, 3) → 8

**• SQRT Function :**

Syntax: SQRT(num)

**Eg:**

1. SQRT(49) → 7

**• RAND Function :**

Syntax: RAND()

**Eg:**

1. RAND() → 0.68

- Generates Random Number

**• COALESCE Function :**

Syntax: COALESCE(value1, value2, value3, ...)

**Eg:**

1. SELECT COALESCE(NULL, 'Nishant') AS result;

*Result*

-------

Nishant

- First value is NULL, so it takes the next one.

1. SELECT COALESCE(NULL, NULL, 'Rajat', 'Nishant') AS result;

*Result*

--------

Rajat

- It picks the first non-NULL value it finds.

| 1. Suppose we have table   **first\_name** | **middle\_name** | **last\_name** |
| --- | --- | --- |
| Nishant | NULL | Kumar |
| Rajat | Pratap | Chaudhary |
| Ankit | NULL | NULL |

SELECT

COALESCE(middle\_name, 'No Middle Name') AS middle\_name\_fixed

FROM students;

*Result*

--------

| **middle\_name\_fixed** |
| --- |
| No Middle Name |
| Pratap |
| No Middle Name |

1. SELECT

COALESCE(email, phone, 'No Contact') AS contact\_info

FROM users;

- If email is NULL, it tries phone, if both NULL → returns 'No Contact'.

1. SELECT

COALESCE(salary, 0) AS final\_salary

FROM employees;

*Result*

-------

| **salary** | **final\_salary** |
| --- | --- |
| 50000 | 50000 |
| NULL | 0 |

1. SELECT

COALESCE(bonus, 0) + salary AS total\_income

FROM employees;

- Prevents your sum from becoming NULL when bonus is missing.