Important SQL Functions

• **<u>SUBSTRING</u> 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
```

2.) *SELECT SUBSTRING('COMPUTER SCIENCE', 10, 7) AS result;*
Result
```
------
SCIENCE
```

3.)
```
SELECT
   email,
   SUBSTRING(email, CHARINDEX('@', email) + 1, LEN(email)) AS domain
FROM students;
```
Result
------

email	domain
nishant@gmail.com	gmail.com
rajat@yahoo.com	yahoo.com
user@iitm.ac.in	iitm.ac.in

4.) *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

emp_code	**last_digits**
EMP56789	6789
EMP99999	9999

• **<u>CASE</u> 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

2.)
```
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

```
3.)  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

• **<u>REPLACE</u> 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!
```

2.) *SELECT REPLACE('123-456-7890', '-', '') AS clean_number;*

3.) *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

4.) *UPDATE students*
SET address = REPLACE(address, 'Collage', 'College');

- Updates all rows where “Collage” appears.

• **<u>CHARINDEX</u> Function :**

Syntax: *CHARINDEX(substring, string, [start_position])*

Eg:

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

3

2.) *SELECT CHARINDEX('z', 'Nishant') AS position;*
Result

0

3.) *SELECT CHARINDEX('i', 'Nishant', 2) AS position;*
Result

0

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

4.) *SELECT*
email,
SUBSTRING(email, 1, CHARINDEX('@', email) - 1) AS username

FROM students;
Result
----

email	username
nishant@gmail.com	nishant
rajat@yahoo.com	rajat

5.) *SELECT SUBSTRING('New Delhi', 1, CHARINDEX(' ', 'New Delhi') - 1) AS first_word;*
Result

New

6.) *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.

• **CONCAT 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

2.) *SELECT CONCAT(first_name, ' ', last_name) AS full_name*
FROM students;

3.) *SELECT CONCAT('Order ID: ', order_id, ', Amount: ₹', total) AS summary*
FROM orders;
Result

Order ID: 101, Amount: ₹1200
Order ID: 102, Amount: ₹500

4.) *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.

5.) *SELECT CONCAT('Hello ', NULL, ' Nishant');*
- In MySQL / PostgreSQL / Oracle → 'Hello Nishant' (NULL ignored)
- In SQL Server → NULL (because NULL makes the whole string NULL)

• **<u>TRIM, LTRIM, and RTRIM</u> Function :**

Syntax: *TRIM([characters] FROM string)*

Eg:

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

Hello

2.) *SELECT TRIM('   Nishant   ') AS cleaned;*
Result

Nishant

3.) *SELECT RTRIM('Nishant   ') AS result;*
Result

Nishant

4.) *SELECT LTRIM(RTRIM('   Nishant   ')) AS cleaned;*
Result

Nishant

5.) *SELECT REPLACE(TRIM(name), ' ', ' ') AS fixed_name*
FROM students;
- Removes leading/trailing spaces, then replaces double spaces with single.

• **<u>ROUND</u> Function :**

Syntax: *ROUND(num, d)*

Eg:

1.) *ROUND(12.3456, 2)* → 12.35

• **<u>CEIL / CEILING</u> Function :**

Syntax: *CEIL(num)*

Eg:

1.) *CEIL(4.2)* → 5

• **<u>FLOOR</u> Function :**

Syntax: *FLOOR(num)*

Eg:

1.) *FLOOR(4.9)* → 4

• **<u>ABS</u> Function :**

Syntax: *ABS(num)*

Eg:

1.) *ABS(-5)* → 5

• **<u>POWER</u> Function :**

Syntax: *POWER(a,b)*

Eg:

1.) *POWER(2, 3)* → 8

• **<u>SQRT</u> Function :**

Syntax: *SQRT(num)*

Eg:

1.) *SQRT(49)* → 7

• **<u>RAND</u> Function :**

Syntax: *RAND()*

Eg:

1.) *RAND()* → 0.68
- Generates Random Number

• **<u>COALESCE</u> 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.

2.) *SELECT COALESCE(NULL, NULL, 'Rajat', 'Nishant') AS result;*
Result

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

3.) 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

4.)
```
SELECT
   COALESCE(email, phone, 'No Contact') AS contact_info
FROM users;
```
- If email is NULL, it tries phone, if both NULL → returns 'No Contact'.

5.)
```
SELECT
   COALESCE(salary, 0) AS final_salary
FROM employees;
```
Result

----

salary	final_salary
50000	50000
NULL	0

6.)
```
SELECT
    COALESCE(bonus, 0) + salary AS total_income
FROM employees;
```
- Prevents your sum from becoming NULL when bonus is missing.