**Function**

Multi Row Function

Single Row Function

Character Manipulation Function

SUM( ) Function

* AVG( ) Function.

Number Manipulation Function

* MIN( ) Function.
* MAX( ) Function.

Data Manipulation Function

* COUNT( ) Function.

Conversion Manipulation Function

General Manipulation Function

* **Single Row functions** - Single row functions are the one who work on single row and return one output per row.
* **Multiple Row functions** - Multiple row functions work upon group of rows and return one result for the complete set of rows. We have to use Group By for multi row functions.

**Character Manipulation Function(Case)**

* **Use of Lower( ) :** //converts into Lower case

Select first\_name, lower (first\_name)

from employees

* **Use of Upper( ) :** //converts into Upper case

Select first\_name, upper (first\_name)

from employees

* **Use of Initcap( ) :** //converts into first letter of each word in uppercase, all other letters in lowercase

Select first\_name, initcap (first\_name)

from employees

* **Search Using Character Manipulation Function (Lower( ) , Upper( ) , Initcap( ) )**

Search employees, where first name starts with ‘david’ :

Select first\_name, lower (first\_name) //You can use Upper & Initcap instead of lower

from employees

where lower(first\_name)= lower('david')

**Character Manipulation Function(Character)**

* **Use of Length( ) :**

Select first\_name, Length (first\_name) //represent length of 1st name

from employees

* **Use of INSTR( ) :**

Select first\_name, INSTR(first\_name, 'a') //find the substring ‘a’ in employees 1st name

from employees

* **Use of LPAD( ) :** //LPAD(string, length, lpad\_string)

The LPAD() function left-pads a string with another string, to a certain length.

Select first\_name, LPAD(salary, 10, '#') //represent salary in 10 digits using ‘#’

from employees

where lower(first\_name)=lower(:input\_value)

* **Use of Concat( ) :**

Select first\_name, Concat(first\_name, last\_name) //combines first & last name

from employees

* Find first name & name of employees who have ‘an’ in their name :

Select first\_name, Concat(first\_name, last\_name)

from employees

where first\_name LIKE '%an' or last\_name LIKE '%an'

* **Use of Substr( ) :**

Select first\_name, last\_name, job\_id//seperates string to search using position

from employees

where SUBSTR(job\_id, 4)='REP' //here ‘REP’ string, is in 4th position as ‘SA\_REP’

* **Use of RPAD( ) :** //RPAD(string, length, rpad\_string)

The RPAD() function right-pads a string with another string, to a certain length.

SELECT RPAD(First\_Name, 10, 'a')

FROM employees

* **Use of TRIM( ) :**

1. SELECT TRIM('@$ ' FROM ' @Whatever$ ')

2. SELECT TRIM('dav')

from employees

* **Use of REPLACE( ) :**

SELECT REPLACE('david', 'd', 'v')

from employees

**Number Function**

|  |  |
| --- | --- |
| 1 | [ABS()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_abs)  Returns the absolute value of numeric expression. |
| 2 | [ACOS()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_acos)  Returns the arccosine of numeric expression. Returns NULL if the value is not in the range -1 to 1. |
| 3 | [ASIN()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_asin)  Returns the arcsine of numeric expression. Returns NULL if value is not in the range -1 to 1 |
| 4 | [ATAN()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_atan)  Returns the arctangent of numeric expression. |
| 5 | [ATAN2()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_atan2)  Returns the arctangent of the two variables passed to it. |
| 6 | [BIT\_AND()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_bit_and)  Returns the bitwise AND all the bits in expression. |
| 7 | [BIT\_COUNT()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_bit_count)  Returns the string representation of the binary value passed to it. |
| 8 | [BIT\_OR()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_bit_or)  Returns the bitwise OR of all the bits in the passed expression. |
| 9 | [CEIL()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_ceil)  Returns the smallest integer value that is not less than passed numeric expression |
| 10 | [CEILING()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_ceiling)  Returns the smallest integer value that is not less than passed numeric expression |
| 11 | [CONV()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_conv)  Convert numeric expression from one base to another. |
| 12 | [COS()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_cos)  Returns the cosine of passed numeric expression. The numeric expression should be expressed in radians. |
| 13 | [COT()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_cot)  Returns the cotangent of passed numeric expression. |
| 14 | [DEGREES()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_degrees)  Returns numeric expression converted from radians to degrees. |
| 15 | [EXP()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_exp)  Returns the base of the natural logarithm (e) raised to the power of passed numeric expression. |
| 16 | [FLOOR()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_floor)  Returns the largest integer value that is not greater than passed numeric expression. |
| 17 | [FORMAT()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_format)  Returns a numeric expression rounded to a number of decimal places. |
| 18 | [GREATEST()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_greatest)  Returns the largest value of the input expressions. |
| 19 | [INTERVAL()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_interval)  Takes multiple expressions exp1, exp2 and exp3 so on.. and returns 0 if exp1 is less than exp2, returns 1 if exp1 is less than exp3 and so on. |
| 20 | [LEAST()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_least)  Returns the minimum-valued input when given two or more. |
| 21 | [LOG()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_least)  Returns the natural logarithm of the passed numeric expression. |
| 22 | [LOG10()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_log10)  Returns the base-10 logarithm of the passed numeric expression. |
| 23 | [MOD()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_mod)  Returns the remainder of one expression by diving by another expression. |
| 24 | [OCT()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_oct)  Returns the string representation of the octal value of the passed numeric expression. Returns NULL if passed value is NULL. |
| 25 | [PI()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_pi)  Returns the value of pi |
| 26 | [POW()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_pow)  Returns the value of one expression raised to the power of another expression |
| 27 | [POWER()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_power)  Returns the value of one expression raised to the power of another expression |
| 28 | [RADIANS()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_radians)  Returns the value of passed expression converted from degrees to radians. |
| 29 | [ROUND()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_round)  Returns numeric expression rounded to an integer. Can be used to round an expression to a number of decimal points |
| 30 | [SIN()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_round)  Returns the sine of numeric expression given in radians. |
| 31 | [SQRT()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_sqrt)  Returns the non-negative square root of numeric expression. |
| 32 | [STD()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_std)  Returns the standard deviation of the numeric expression. |
| 33 | [STDDEV()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_stddev)  Returns the standard deviation of the numeric expression. |
| 34 | [TAN()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_tan)  Returns the tangent of numeric expression expressed in radians. |
| 35 | [TRUNCATE()](https://www.tutorialspoint.com/sql/sql-numeric-functions.htm#function_truncate)  Returns numeric exp1 truncated to exp2 decimal places. If exp2 is 0, then the result will have no decimal point. |

* **Use of ROUND( ) :**

Select ROUND(23.83921, 2) //it will show output 23.84

From DUAL //DUAL is a dummy table, where raw values are inserted

**Date Functions**

**SYS\_DATE :**

Select employee\_id, hire\_date, sysdate //shows system date

From employees

ROUND and TRUNC functions are used to round and truncates the date value.

**MONTHS\_BETWEEN( ):**

Select employee\_id, hire\_date,sys\_date, round(MONTHS\_BETWEEN (sys\_date,hire\_date)/12)

From employees

**ADD\_MONTHS( ) :**

Select employee\_id, hire\_date,sys\_date, add\_months(sys\_date,3)

From employees

**NEXT\_DAY( ):**

Select employee\_id, hire\_date,sys\_date, next\_day(sys\_date,sunday)

From employees

**LAST\_DAY( ):**

Select employee\_id, hire\_date,sys\_date, last\_day(sys\_date,sunday)

From employees

**Conversion Functions**

**Use of TO\_CHAR():**

**Represent date in 'DD/MM/YYYY' format -**

Select employee\_id, hire\_date, sysdate, TO\_CHAR(sysdate,'DD/MM/YYYY')

From employees

**Represent Hire Date & Salary in** ‘**$99999.99’ format -**

Select employee\_id, hire\_date, TO\_CHAR (salary, '$99999.99')

From employees

**Use of TO\_NUMBER():**

1. SELECT TO\_NUMBER('224.21', '9G999D99')

FROM DUAL

1. SELECT TO\_NUMBER('2254.21', '9999.99')

FROM DUAL;

**Use of TO\_DATE():**

SELECT TO\_DATE('January 15, 1989', 'Month dd, YYYY')

FROM DUAL;

**General Functions**

**Use of NVL():** Syntax : NVL(C\_N, Value) , C\_N is column name //2 expressions

SELECT first\_name, NVL(JOB\_ID, 'n/a')

FROM employees;

**Use of NVL2( ):** Syntax :NVL2( string1, value\_if\_not\_null, value\_if\_null ) //3 expressions

1. Show Employee who gets salary with commissions & without commissions :

SELECT employee\_id,commission\_pct,salary, NVL2(commission\_pct, salary, salary+salary\*commission\_pct)

FROM employees;

**Use of COALESCE( ):** Syntax : COALESCE (expr1, expr2, ... expr\_n ) //n>3 expressions

SELECT first\_name, last\_name, COALESCE (Salary, commission\_pct) AS SALARY\_COMMISSION

FROM employees

**Use of DECODE( ):**

SELECT first\_name, salary, DECODE (hire\_date, sysdate,'NEW JOINEE','EMPLOYEE')

FROM employees;

**Use of CASE :** //When….Then works as If …. Else

1. SELECT first\_name, CASE WHEN salary < 200 THEN 'GRADE 1'

WHEN salary > 200 AND salary < 5000 THEN 'GRADE 2'

ELSE 'GRADE 3'

END CASE

FROM employees

1. Select employee\_id , job\_id ,

CASE job\_id WHEN 'IT\_PROG' THEN salary+salary\*0.1

WHEN 'SA\_REP' THEN salary+salary\*0.15

ELSE salary END

from Employees

**Multi Row Function**

**Types of Multi-Row Functions :**

* Maximum(Max)
* Minimum(MIN)
* Average(Avg)
* Sum
* Count

The **Group By** is used to group data based on the same value in a specific column. The **ORDER BY** sorts the result and shows it in ascending or descending order.

* SELECT job\_id, sum(salary), round(avg(salary)), max(salary), min(salary), count(job\_id)

FROM employees

group by job\_id

The GROUP BY statement is often used with aggregate functions ( COUNT( ),  MAX( ),  MIN( ), SUM( ),  AVG( ) ) to group the result-set by one or more columns.

**Join**

In DBMS, a join statement is mainly used to combine two tables based on a specified common field between them.

2 types of join;

1. Inner Join – Natural & Equi
2. Outer Join – Left, Right & Full

**Inner Join**

A join that can be used to return all the values that have matching values in both the tables.

**Syntax:**

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching\_column = table2.matching\_column;

**Natural Join :**

Select \*

from Employees natural join Departments

**Equi Join :**

Select \*

from Employees e ,Departments d

where e.department\_id=d.department\_id

**Outer Join**

Outer Join is a join that can be used to return the records in both the tables whether it has matching records in both the tables or not.

**Right Join Syntax:**

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching\_column = table2.matching\_column;

* Select \*

from Employees e right outer join Departments d

on e.department\_id=d.department\_id

**Left Join Syntax:**

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching\_column = table2.matching\_column;

* Select \*

from Employees e left outer join Departments d

on e.department\_id=d.department\_id

**Full Join Syntax:**

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

FULL JOIN table2

ON table1.matching\_column = table2.matching\_column;

* Select \*

from Employees e full outer join Departments d

on e.department\_id=d.department\_id

**Sub Query**

A subquery is a SQL query nested inside a larger query.

* Occurs in :
  + A SELECT clause
  + A FROM clause
  + A WHERE clause

1. SELECT \*

FROM Employees

WHERE department\_id = (SELECT department\_id

FROM employees

WHERE first\_name='Lex') and first\_name!='Lex'

1. SELECT a.studentid, a.name, b.total\_marks

FROM student a, marks b

WHERE a.studentid = b.studentid AND b.total\_marks > (SELECT total\_marks

FROM marks

WHERE student\_id = 'V002' )

1. SELECT \*

FROM Employees

WHERE department\_id IN (SELECT department\_id

FROM Employees

WHERE first\_name='David')