**SQL FUNCTIONS:**  
SQL functions are built into Oracle Database and are available for use in various appropriate SQL statements. The essential capabilities of a function can be the case conversion of strings, in-string or substring operations, mathematical computations on numeric data, and date operations on date type values. SQL Functions optionally take arguments from the user and mandatorily return a value. In short,  
SQL functions can:

1. Perform calculations on data
2. Modify individual data items
3. Manipulate output for groups of rows
4. Format dates and numbers for display
5. Convert column data types.

**Single Row functions** - Single row functions are the one who work on single row and return one output per row. This function can manipulate data items, accept arguments and return one value, act on each row returned, return one result per row, may modify the data type, can be nested and accept argument which can be a column or an expression

1. ***Case Conversion function*:** These functions convert between different data types. It includes

a) Lower: - LOWER returns char, with all letters lowercase. It can take column name and a string as an argument.   
Ex:  
SELECT empno, LOWER (ename)

FROM emp;

b) Upper: - UPPER returns char, with all letters uppercase. It can take column name and a string as an argument.   
Ex:

SELECT empno, UPPER (ename)

FROM emp;

c) Initcap: - INITCAP returns char, with the first letter of each word in uppercase, all other letters in lowercase. Words are delimited by white space or characters that are not alphanumeric.  
Ex:

Select initcap(ename), initcap(job) from emp where ename = ‘KING’;

1. ***Character manipulation functions****:* Character functions operate on values of character class data type, i.e., Char, Varchar2, Varchar etc. These functions can return either character class data type or number class data type based on the operation performed on the input data.

a) SUBSTR: The SUBSTR functions return a portion of char, beginning at character position, substring\_length characters long. It can be written as substr(char, position,substing\_length)..  
 If position is positive, then Oracle Database counts from the beginning of char to find the first character.  
 if position is negative, then Oracle counts backward from the end of char.

b) Length: The LENGTH functions return the length of char. LENGTH calculates length using characters as defined by the input character set.

c) LPAD|RPAD: LPAD and RPAD functions pad the given string up to a specific length with a given character.

d) LTRIM|RTRIM: TRIM function trims the string input from the start or end.

e) REPLACE: REPLACE function replaces characters from the input string with a given character.

f) CONCAT: CONCAT function concatenates two string values.

3)  ***Numeric function*:** Numeric functions accept numeric input and return numeric values. Most numeric functions that return NUMBER values that are accurate to 38 decimal digits.

a) Round**:** The ROUND() function is used to round a numeric field to the number of decimals specified. Syntax: SELECT ROUND(column\_name,decimals) FROM table\_name;  
column name = any column and decimal= 0,1,2……..

Ex: SELECT ProductName, ROUND(Price,0) AS RoundedPrice  
FROM Products;

b) Trunc:The TRUNC (number) function returns n1 truncated to n2 decimal places. If n2 is omitted, then n1 is truncated to 0 places. n2 can be negative to truncate (make zero) n2 digits left of the decimal point.  
The following examples truncate numbers:

SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;

Truncate

----------

15.7

SELECT TRUNC(15.79,-1) "Truncate" FROM DUAL;

Truncate

----------

10

4) ***Date function***: These functions accept date value parameters.

**a**) months\_between**:** MONTHS\_BETWEEN function returns the count of months between the two dates.

b) add\_months**:** ADD\_MONTHS function add 'n' number of months to an input date.

c) next\_day**:** NEXT\_DAY function returns the next day of the date specified.

d) to\_date:TO\_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE datatype.

**5) *Other function:***

a) NVL: NVL lets you replace null (returned as a blank) with a string in the results of a query. If expr1 is null, then NVL returns expr2. If expr1 is not null, then NVL returns expr1.  
Generally, denoted by: NVL (exp1,exp2)

b) NVL2: NVL2 lets you determine the value returned by a query based on whether a specified expression is null or not null. If expr1 is not null, then NVL2 returns expr2. If expr1 is null, then NVL2 returns expr3.  
Generally, denoted by: NVL2(exp1,exp2,exp3)

c) DECODE:DECODE compares expr to each search value one by one. If expr is equal to a search, then Oracle Database returns the corresponding result. If no match is found, then Oracle returns default. If default is omitted, then Oracle returns null.  
Denoted by**:** Decode (expr, value1, result1, value2, result2, default)

**6) *TO CHAR*:**  The following is a list of valid parameters when the **TO\_CHAR function** is used to convert a date to a string.

|  |  |
| --- | --- |
| **Parameter** | **Explanation** |
| YEAR | Year, spelled out |
| YYYY | 4-digit year |
| YYY YY Y | Last 3, 2, or 1 digit(s) of year. |
| MM | Month (01-12; JAN = 01). |
| MON | Abbreviated name of month. |
| MONTH | Name of month, padded with blanks to length of 9 characters. |
| D | Day of week (1-7). |
| DAY | Name of day. |
| DD | Day of month (1-31). |
| DDD | Day of year (1-366). |
| DY | Abbreviated name of day. |
| HH | Hour of day (1-12). |
| HH12 | Hour of day (1-12). |
| HH24 | Hour of day (0-23). |
| MI | Minute (0-59). |
| SS | Second (0-59). |

A.M or P.M AM/PM  
R.M Roman numeral month (I-XII; JAN = I).  
BC or AD BC/AD

**Multiple Row functions** - Multiple row functions work upon group of rows and return one result for the complete set of rows. They are also known as Group Functions.

1. **Min:** The **min()** function returns the lowest value of a specified column from a group of rows.
2. **Max:** The **max()** function returns the highest value of a specified column from a group of rows
3. **Sum:** The **sum()** function returns the total of all values for a specified column in a group of rows
4. **Avg:** The **avg()** function returns the average value of a numeric field from a group of rows.
5. **Count:** The **count()** function counts the number of rows in a group of rows. This function counts all rows in the group, including those for which a **NULL** value is present.
6. **Variance:** The Oracle VARIANCE function returns the variance or *variability* of an expression. In other words, VARIANCE computes how much a set of numbers varies or differs within the set.  
     
   The following example calculates the variance of all salaries in the sample employees table:

SELECT VARIANCE(salary) "Variance"

FROM employees;

Variance

----------

15283140.5

1. **Stddev:** STDDEV returns the sample standard deviation of expr, a set of numbers. This function takes as an argument any numeric data type or any nonnumeric data type that can be implicitly converted to a numeric data type. The function returns the same data type as the numeric data type of the argument.  
   EX:

SELECT STDDEV(salary) "Deviation"

FROM employees;

Deviation

----------

3909.36575

**GROUP BY CLAUSE:**  
The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups. Typically a GROUP BY clause is used in conjunction with an aggregate expression (Multiple row functions). It is mandatory that columns referenced in the SQL **SELECT** statement list, except for aggregate expressions, must be included in the **GROUP BY** clause. On the contrary, columns used in group by clause don’t have to be listed in select statement.   
Group by clause can’t take column aliases and the result derived from select statement which includes group by clause is in ascending order.

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

**SUB QUERIES:**   
A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved OR Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value in the database.  
Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE, WHERE, HAVING statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

There are a few rules that subqueries must follow:

1. Subqueries must be enclosed within parentheses.
2. Place the sub query on the right side of the comparison.
3. Use single row operators with single row sub queries and multiple row operators with multiple row sub queries.

**TYPES OF SUBQUERIES:**

1. SINGLE ROW SUBQUERY: Sub queries that can return only one or zero rows to the outer statement are called **single-row sub queries.** Single-row sub queries are sub queries used with a comparison operator ( = ,> , < , <= , >=, <> ) in a WHERE, or HAVING clause.
2. MULTIPLE ROW SUB QUERY: Sub queries that can return more than one row (but only one column) to the outer statement are called **multiple-row sub queries**. Multiple-row sub queries are sub queries used with an IN, ANY, or ALL clause.
3. Using IN operator with a Multiple Row Sub query: IN operator is used to check a value within a set of values. It indicates that the record processed by the outer queries must match one of the values returned by the inner query.
4. Using ANY operator with a Multiple Row Sub query: The ANY operator (and its synonym SOME operator) compares a value to each value returned by a subquery.  
   You must place an = (equivalent to IN operator), > (records that have a value greater than the lowest value returned by the sub query.), < (records that have a value less that the highest value returned by the sub query.) operator before ANY in your query.
5. Using ALL operator with a Multiple Row Sub query: The ALL operator compares value to every value returned by a sub query. >ALL means more than the maximum and <ALL means less than the minimum
6. MULTIPLE COLUMN SUB QUERY: If you want to compare two or more columns, you must write a compound WHERE clause using logical operators. Multiple-column subqueries enable you to combine duplicate WHERE conditions into a single WHERE clause. Multiple column sub queries return multiple columns. Multiple column sub queries can be pairwise or non-pairwise.