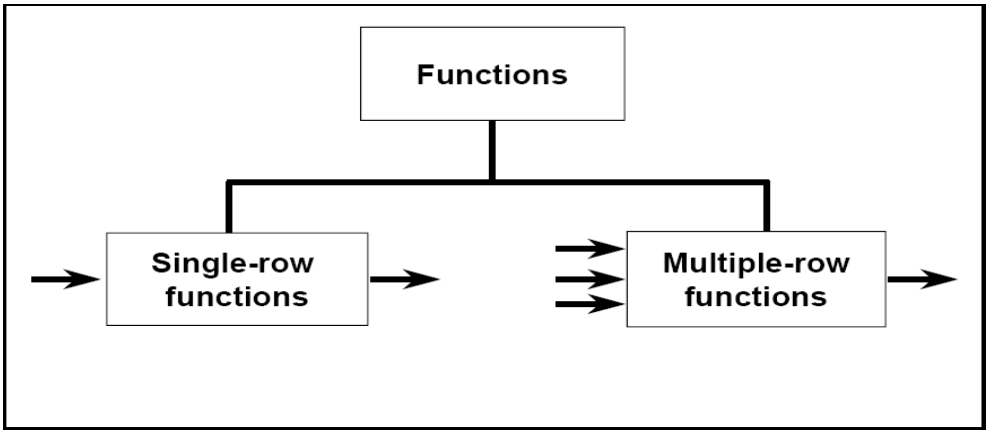
NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES
CL 203-Database Systems
Lab Session 02

ORACLE BUILT-IN FUNCTIONS

Functions are a very powerful feature of SQL and can be used to do the following:

- Perform calculations on data
- Modify individual data items
- Manipulate output for groups of rows
- Format dates and numbers for display
- Convert column data types

There are two distinct types of functions:
- Single-row functions
- Multiple-row functions/Group Functions/Aggregate Functions



There are four types of single row functions. They are:

1) Numeric Functions
These are functions that accept numeric input and return numeric values.

2) Character or Text Functions
These are functions that accept character input and can return both character and number values.

3) Date Functions
These are functions that take values that are of datatype DATE as input and return values of datatype DATE, except for the MONTHS_BETWEEN function, which returns a number.

4) Conversion Functions
These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

You can combine more than one function together in an expression. This is known as nesting of functions.

Numeric Functions

Numeric functions are used to perform operations on numbers. They accept numeric values as input and return numeric values as output. Few of the Numeric functions are:

Function Name	Return Value
ABS (x)	Absolute value of the number '*x*'
CEIL (x)	Integer value that is Greater than or equal to the number '*x*'
FLOOR (x)	Integer value that is Less than or equal to the number '*x*'
TRUNC (x, y)	Truncates value of number '*x*' up to '*y*' decimal places
ROUND (x, y)	Rounded off value of the number '*x*' up to the number '*y*' decimal places

The following examples explains the usage of the above numeric functions

Function Name	Examples	Return Value
ABS (x)	ABS (1) ABS (-1)	1 -1
GREATEST(value1, value2, …)	GREATEST(7,8,10)	10
LEAST(value1, value2, …)	LEAST(7,8,10)	7
CEIL (x)	CEIL (2.83) CEIL (2.49) CEIL (-1.6)	3 3 -1
FLOOR (x)	FLOOR (2.83) FLOOR (2.49) FLOOR (-1.6)	2 2 -2
TRUNC (x, y)	ROUND (125.456, 1) ROUND (125.456, 0) ROUND (124.456, -1)	125.4 125 120
ROUND (x, y)	TRUNC (140.234, 2) TRUNC (-54, 1) TRUNC (5.7) TRUNC (142, -1)	140.23 54 5 140

Character Functions:

Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Few of the character or text functions are as given below:

Function Name	Return Value
LOWER (string_value)	All the letters in *'string_value'* is converted to lowercase.
UPPER (string_value)	All the letters in *'string_value'* is converted to uppercase.
INITCAP (string_value)	All the letters in *'string_value'* is converted to mixed case.
LTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the left of *'string_value'*.
RTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the right of*'string_value'* .
TRIM (trim_text FROM	All occurrences of *'trim_text'* from the left and right

string_value)	of *'string_value' ,'trim_text'* can also be only one character long .
SUBSTR (string_value, m, n)	Returns *'n'* number of characters from*'string_value'* starting from the '*m'*position.
LENGTH (string_value)	Number of characters in *'string_value'*in returned.
LPAD (string_value, n, pad_value)	Returns '*string_value'* left-padded with*'pad_value'* . The length of the whole string will be of *'n'* characters.
RPAD (string_value, n, pad_value)	Returns '*string_value'* right-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.

The following examples explains the usage of the above character or text functions

Function Name	Examples	Return Value
LOWER(string_value)	LOWER('Good Morning')	good morning
UPPER(string_value)	UPPER('Good Morning')	GOOD MORNING
INITCAP(string_value)	INITCAP('GOOD MORNING')	Good Morning
LTRIM(string_value, trim_text)	LTRIM ('Good Morning', 'Good)	Morning
RTRIM (string_value, trim_text)	RTRIM ('Good Morning', ' Morning')	Good
TRIM (trim_text FROM string_value)	TRIM ('o' FROM 'Good Morning')	Gd Mrning
SUBSTR (string_value, m, n)	SUBSTR ('Good Morning', 6, 7)	Morning
LENGTH (string_value)	LENGTH ('Good Morning')	12
LPAD (string_value, n, pad_value)	LPAD ('Good', 6, '*')	**Good
RPAD (string_value, n, pad_value)	RPAD ('Good', 6, '*')	Good**

Date Functions

These are functions that take values that are of datatype DATE as input and return values of datatypes DATE, except for the MONTHS_BETWEEN function, which returns a number as output.

Few date functions are as given below.

Function	**Result**	**Description**
MONTHS_BETWEEN('01-SEP-95', '11-JAN-94')	19.6774194	Number of months between two dates
ADD_MONTHS('11-JAN-94', 6)	'11-JUL-94'	Add calendar months to dates
NEXT_DAY('01-SEP-95', 'FRIDAY')	'08-SEP-95'	Next day of the date specified
LAST_DAY('01-SEP-95')	'30-SEP-95'	Last day of the month
ROUND(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'MONTH')	01-AUG-95	Round date
ROUND(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'YEAR')	01-JAN-96	Round date

TRUNC(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'MONTH')	01-JUL-95	Truncate date
TRUNC(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'YEAR')	01-JAN-95	Truncate date

Conversion Functions

These are functions that help us to convert a value in one form to another form. For Ex: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE.

Few of the conversion functions available in oracle are:

Function Name	Return Value
TO_CHAR (x [,y])	Converts Numeric and Date values to a character string value. It cannot be used for calculations since it is a string value.
TO_DATE (x [, date_format])	Converts a valid Numeric and Character values to a Date value. Date is formatted to the format specified by *'date_format'*.
NVL (x, y)	If *'x'* is NULL, replace it with *'y'*. *'x'* and *'y'*must be of the same datatype.
DECODE (a, b, c, d, e, default_value)	Checks the value of *'a'*, if *a = b*, then returns*'c'*. If *a = d*, then returns *'e'*. Else, returns*default_value*.

The below table provides the examples for the above functions

Function Name	Examples	Return Value
TO_CHAR ()	TO_CHAR (3000, '$9999') TO_CHAR (SYSDATE, 'Day, Month YYYY')	$3000 Monday, June 2008
TO_DATE ()	TO_DATE ('01-Jun-08')	01-Jun-08
NVL ()	NVL (null, 1)	1

GROUPING FUNCTIONS (multi-valued functions)

A **group function** is an Oracle SQL function that returns a single result based on many rows, as opposed to single-row functions. These functions are: AVG, COUNT, MIN, MAX, STDDEV, SUM, VARIANCE, etc. Grouping functions may include either of the keywords DISTINCT or ALL.ALL is the default if neither is specified and uses all selected rows in the calculation. DISTINCT uses only one row for each value in the calculation.

Example:

• AVG(ALL 2,2,3,3,4) and AVG(2,2,3,3,4) both return 2.8.

• AVG(DISTINCT 2,2,3,3,4) returns 3

FUNCTIONS	DESCRIPTION
AVG(col-name)	The AVG() Func returns the average value of a numeric column.
MIN(col-name)	The min() func returns the smallest value of the selected column.
MAX(col-name)	The max() func returns the largest value of the selected column.
SUM(col-name)	Returns the total sum of a numeric column.
FIRST(COL-NAME)	Returns the First value of the selected column.
LAST(COL-NAME)	Returns the Last value of the selected column.

Examples

i. To show the average salary, minimum salary, maximum salary and count of employees in the organization

```
SELECT AVG (SAL), MIN(SAL), MAX(SAL),
            COUNT(*) FROM EMP;
```

ii. To show the minimum and maximum hiredate for employees

```
SELECT MIN (hiredate),
MAX(hiredate) FROM emp;
```

iii Compute the difference between the minimum and maximum salary.

```
SELECT MAX(SAL) - MIN(SALFROM EMP;
```

iv. To return the number of rows in a table

```
SELECT COUNT(*)
FROM emp
WHERE deptno = 30;
```

v. The group function like AVG do not include null rows. The NVL function forces group functions to include null values.

```
SELECT AVG(NVL(comm, 0))
FROM emp;
```

The GROUP BY Statement

The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.

SQL GROUP BY Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

Examples

i. To show the department-wise average salary,

```
SELECT deptno, AVG(sal)
AVERAGE_SALARY FROM emp
GROUP BY deptno;
```

Note that all columns in the SELECT list that are not in group functions must be in the
GROUP BY clause.

ii. To show the job-wise total salary for each department
SELECT deptno, job, sum(sal)
FROM emp
GROUP BY deptno, job;

Excluding groups result

In the same way that we use the WHERE clause to restrict the rows that we select, the HAVING clause is used to restrict groups. First the group function is applied and the groups matching the HAVING clause are displayed.

The syntax of the SELECT statement showing the HAVING clause along with the GROUP BY clause is shown below:-

SELECT *column, group_function*
FROM *table*
[WHERE condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column];

The HAVING clause can precede the GROUP BY clause but it is recommended that the
GROUP BY clause come first because it is more logical.

Examples

i. To show the department-wise average and maximum salary, in the descending order of average salary, for all departments having average salary higher than 4500.
SELECT DEPTNO, AVG(SAL),MAX(SAL)
FROM EMP GROUP BY DEPTNO
HAVING AVG(SAL) > 2000 ORDER BY
AVG(SAL)

ii. To display the job title and total monthly salary for each job title with a total payroll exceeding 5000.
SELECT JOB, SUM(SAL)
PAYROLL FROM EMP
WHERE JOB NOT LIKE 'SALES%'
GROUP BY JOB
HAVING SUM(SAL) > 5000
ORDER BY SUM(SAL);

Nesting Group Functions

i. To display the maximum average salary by nesting group functions
SELECT max(avg(sal))
FROM emp
GROUP BY deptno;

ACTIVITY

1. Print an employee name(first letter capital) and job title (lower Case)

2. Display the employee number, name (IN UPPER CASE) and department number for employee Blake

3. Display employee name and job joined together, length of employee name and the numeric position of letter A in employee name, for all employees who are in sales.

4. For all employees employed for fewer than 200 months, display the employee number, hiredate, number of months employed, six-month review date, first Friday after hiredate and last day of the month hired.

5. Comparing the hire dates for all employees who started in 1982, display the employee number, hiredate, and month started using the ROUND and TRUNC functions.

6. To display the employee number, the month number and year of hiring.

7. To display the employee name and hiredate for all employees. The hiredate appears as 17 November, 1981.

8. Print the employee name and time of joining in format HH:MI:SS (Assuming that hiredate column were used for storing joining time)

9. Display the salary of employee SCOTT with $ sign preceded

10. Display the names and hire dates of all the employees who joined on February 22,1981.

11. Write down the result of the function calls in the first column to the second column.

Function Call	**Result**
SUBSTR(CONCAT('HIGH', 'SALARY'), 4, 6)	
CONCAT(SUBSTR('INFORMATION', 3, 4),	
INSTR(CONCAT('GET', 'ING'), 'TIN')	
ROUND(69.476, 1)	
TRUNC('13-MAR-90', 'MONTH')	
TRUNC('13-MAR-90', 'YEAR')	
MOD(90, LENGTH('SEVENTY'))	
MONTHS_BETWEEN('14-AUG-96', '23-MAR-95')	

12. To display the employee number, name, salary, salary increase by 15% expressed as a whole number (labeled as *New Salary*), the difference between old salary and new salary (labeled as *Increment*).

13. To display the employee name and calculate the number of months between today and the date the employee was hired (Labeled as *Months_Worked*). Order the results by the number of months employed and round the number of months up to the closest whole number.

14. Write a query that produces the following for each employee:
    <*employee name*> earns <*salary*> monthly

15. Display the employee's name (labeled *name*) with the first letter capitalized and all other letters lowercase and the length of their name (labeled *length*), for all employees whose name starts with J, A or M.

16. list the name, hiredate, and day of the week (labeled *DAY*) on which job was started. Order the result by day of week starting with Monday.

17. Display the job-wise count of employees in each department as follows:-

    DEPTNO JOB NUM_EMP

18. Display the department name, location name, number of employees and the average salary for all employees in that department. Label the columns DNAME, LOC, NUMBER OF PEOPLE and SALARY, respectively. Round the average salary to two decimal places.

19. Display the difference between the highest and lowest salaries. (Labeled as *DIFFERENCE*)

BEST OF LUCK