# CHAPTER 7

1. **Single-row function**

These functions operate on single rows only and return one result per row. There are different types of single-row functions. This lesson covers the following ones:

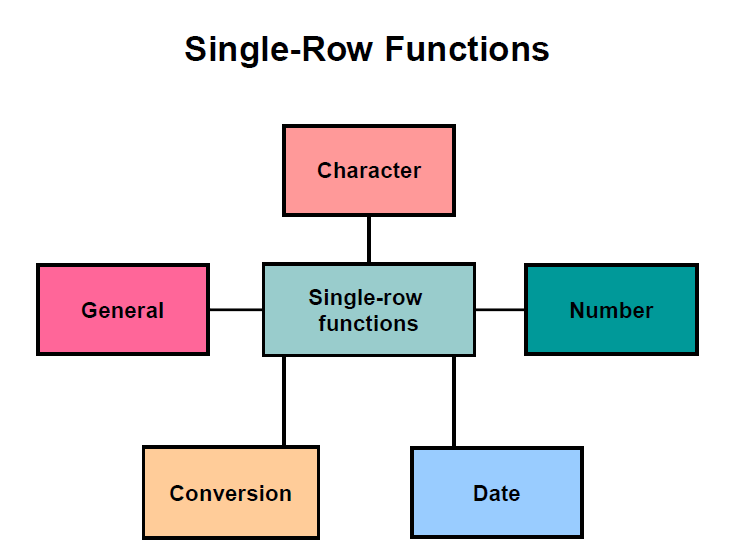
• Character

• Number

• Date

• Conversion

• General



**Character functions**: Accept character input and can return both character and

number values

• **Number functions**: Accept numeric input and return numeric values

• **Date functions**: Operate on values of the DATE data type (All date functions return a value of DATE data type except the MONTHS\_BETWEEN function, which returns a

number.)

• **Conversion functions**: Convert a value from one data type to another

• **General functions:**

- NVL

- NVL2

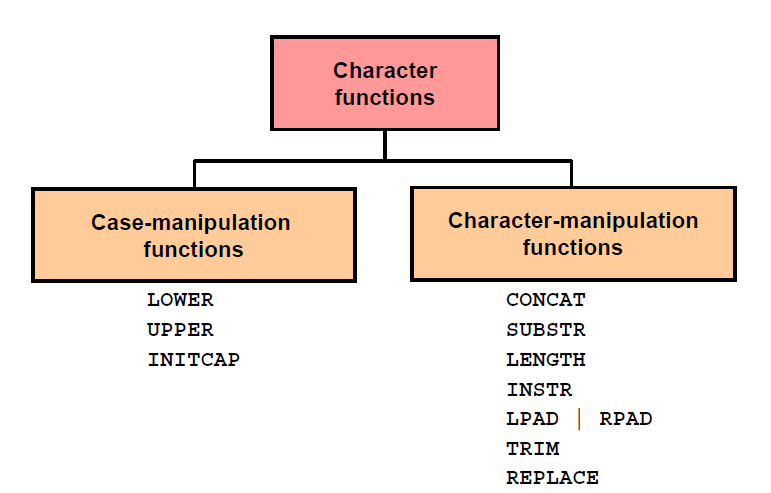
- NULLIF

- COALESCE

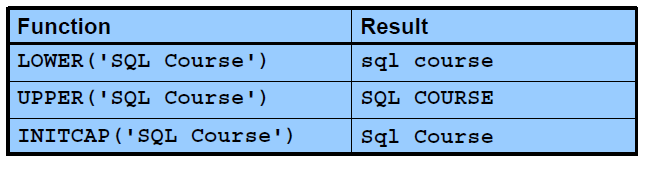
- CASE

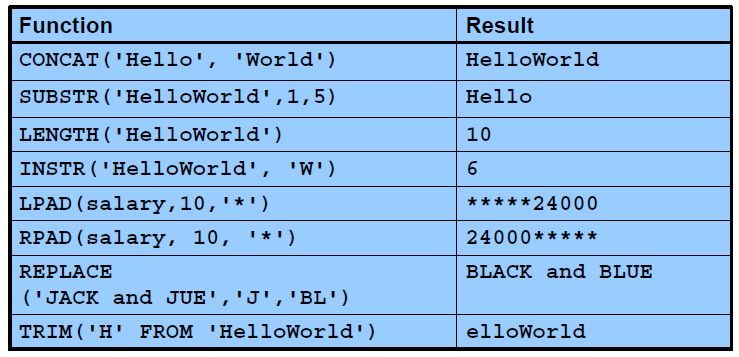
- DECODE

a) Character



Example:





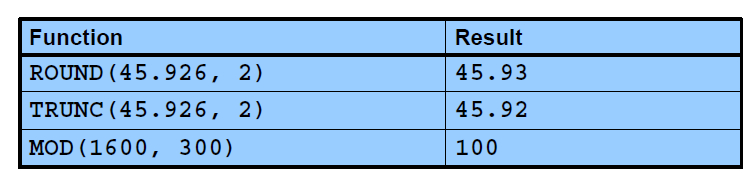
b) Number

ROUND: Rounds value to specified decimal

TRUNC: Truncates value to specified decimal

MOD: Returns remainder of division

Example:



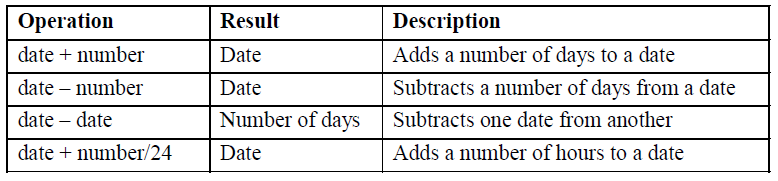
c) Date

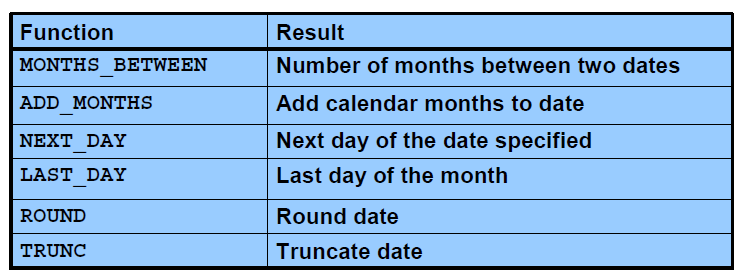
The Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, and seconds.

The default date display format is DD-MON-RR

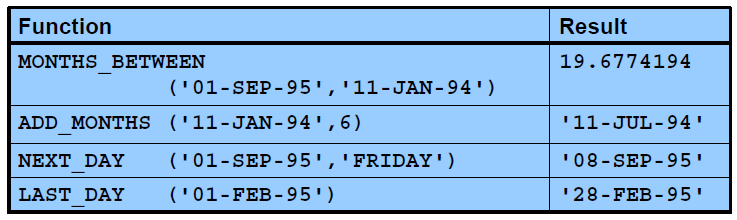
SYSDATE is a function that returns: date and time

You can perform the following operations:

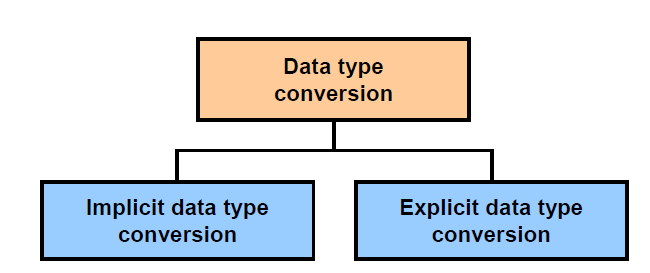




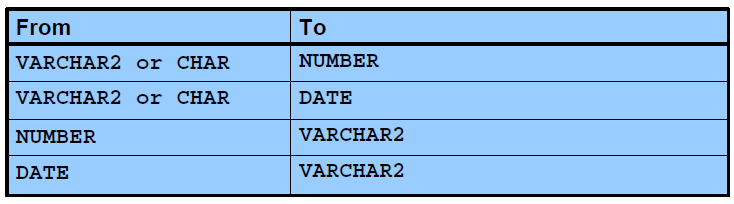
Example:



d) Conversion



* Implicit Data



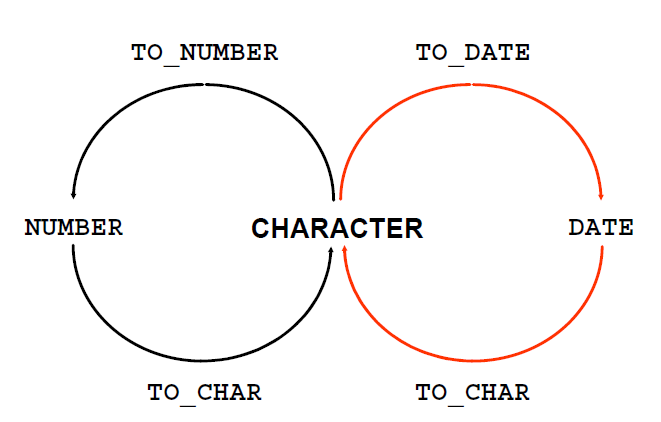
Example:

**select** \* **from** employees **where** hire\_date > '01-JAN-1990'

**select** \* **from** employees **where** salary = 24000

**select** \* **from** employees **where** salary = '24000'

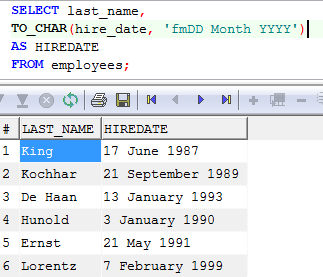
* Explicit Data



Example:

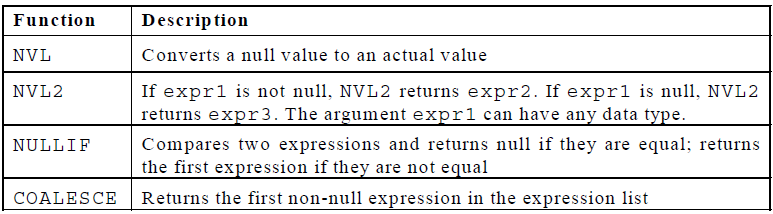
**SELECT** employee\_id, **TO\_CHAR**(hire\_date, 'MM/YY') Month\_Hired

**FROM** employees **WHERE** last\_name = 'Higgins';

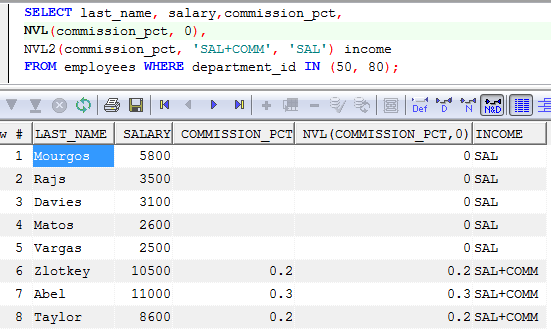


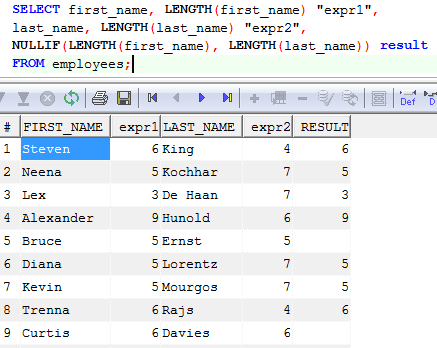
e) General

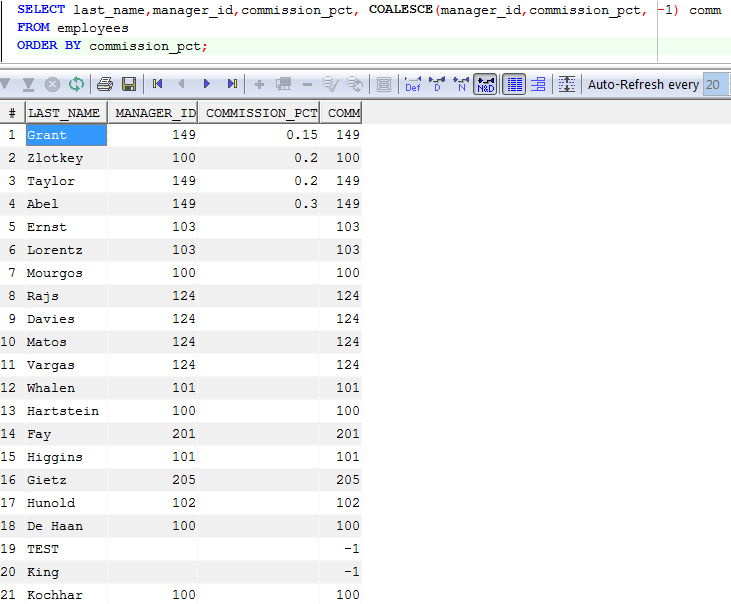
These functions work with any data type and pertain to the use of null values in the expression list.



Example:

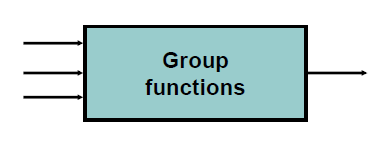






1. **Multiple-row function**

Functions can manipulate groups of rows to give one result per group of rows. These functions are also known as *group functions: Avg, count, max, min, sum*



Syntax:

**SELECT *column*, *group\_function***

**FROM *table***

**[WHERE *condition*]**

**[GROUP BY *group\_by\_expression*]**

**[HAVING *group\_condition*]**

**[ORDER BY *column*];**

In the syntax:

*group\_by\_expression* specifies columns whose values determine the basis for grouping rows

*group\_condition* restricts the groups of rows returned to those groups for which the specified condition is true.

1. **User-defined function**

A **stored function** (also called a **user function** or **user-defined function**) is a set of PL/SQL statements you can call by name. Stored functions are very similar to procedures, except that a function returns a value to the environment in which it is called. User functions can be used as part of a SQL expression

* Syntax:

**CREATE [OR REPLACE] FUNCTION *function\_name***

**[(*argument1* [*mode1*] *datatype1,***

***argument2* [*mode2*] *datatype2,***

***. . .*)]**

**RETURN *datatype***

**IS|AS**

**function\_body;**

**In syntax:**

*function\_name* Is the name of the function to be created

*argument* Is the name given to the function parameter (Every argument is associated with a mode and data type. You can have any number of arguments separated by a comma. You pass the argument when you invoke the function.)

*mode* Is the type of parameter (Only IN parameters should be declared.)

*datatype* Is the data type of the associated parameter

RETURN *datatype* Is the data type of the value returned by the function

*function\_body* Is the PL/SQL block that makes up the function code

* Example

**CREATE** **or** **Replace** **FUNCTION** check\_sal(p\_empno employees.employee\_id%**TYPE**)

**RETURN** **Boolean** **IS**

v\_dept\_id employees.department\_id%**TYPE**;

v\_sal employees.salary%**TYPE**;

v\_avg\_sal employees.salary%**TYPE**;

**BEGIN**

**SELECT** salary,department\_id **INTO** v\_sal,v\_dept\_id **FROM** employees

**WHERE** employee\_id=p\_empno;

**SELECT** **avg**(salary) **INTO** v\_avg\_sal **FROM** employees

**WHERE** department\_id=v\_dept\_id;

**IF** v\_sal > v\_avg\_sal **THEN**

**RETURN** **TRUE**;

**ELSE**

**RETURN** **FALSE**;

**END** **IF**;

**EXCEPTION** **when** others **then** **return** **false**;

**end**;

Invoke

**BEGIN**

DBMS\_OUTPUT.PUT\_LINE('Checking for employee with id 205');

**IF** (check\_sal(205) **IS** **NULL**) **THEN**

DBMS\_OUTPUT.PUT\_LINE('The function returned

NULL due to exception');

**ELSIF** (check\_sal(205)) **THEN**

DBMS\_OUTPUT.PUT\_LINE('Salary > average');

**ELSE**

DBMS\_OUTPUT.PUT\_LINE('Salary < average');

**END** **IF**;

**END**;

1. **Practices**
   1. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS\_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.
   2. Create a report that produces the following for each employee:

<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

* 1. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the $ symbol. Label the column SALARY.
  2. Display each employee’s last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to “Monday, the Thirty-First of July, 2000.”
  3. Create a query that displays the employees’ last names and commission amounts. If an employee does not earn commission, show “No Commission.” Label the column COMM.
  4. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number
  5. Write a query to display the number of people with the same job.
  6. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.
  7. Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998
  8. Create a function to display all employee of one department.