**EXERCISE-5**

**In built Functions**

**Single row functions:**

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.

Accept arguments which can be a column or an expression

**Syntax**

Function\_name(arg1,…argn)

An argument can be one of the following

* User-supplied constant
* Variable value
* Column name
* Expression

**SINGLE-ROW FUNCTIONS**

**CONVERSION**

**DATE**

**NUMBER**

**CHARACTER**

**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.
* Conversion Functions: Convert a value from one type to another.

**Character Functions**

Character Functions

Case-manipulation functions Character-manipulation functions

1. Lower 1. Concat
2. Upper 2. Substr
3. Initcap 3. Length

4. Instr

5. Lpad/Rpad

6. Trim

7. Repalce

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| lower(column/expr) | Converts alpha character values to lowercase |
| upper(column/expr) | Converts alpha character values to uppercase |
| initcap(column/expr) | Converts alpha character values the to uppercase for the first letter of each word, all other letters in lowercase |
| concat(column1/expr1, column2/expr2) | Concatenates the first character to the second character |
| substr(column/expr,m,n) | Returns specified characters from character value starting at character position m, n characters long |
| length(column/expr) | Returns the number of characters in the expression |
| instr(column/expr,’string’,m,n) | Returns the numeric position of a named string |
| lpad(column/expr, n,’string’) | Pads the character value right-justified to a total width of n character positions |
| rpad(column/expr,’string’,m,n) | Pads the character value left-justified to a total width of n character positions |
| trim(leading/trailing/both, trim\_character FROM trim\_source) | Enables you to trim heading or string. trailing or both from a character |
| replace(text, search\_string, replacement\_string) |  |

**Example:**

lower(‘SQL Course’)🡪sql course

upper(‘SQL Course’)🡪SQL COURSE

initcap(‘SQL Course’)🡪Sql Course

SELECT ‘The job id for’|| upper(last\_name||’is’||lower(job\_id) AS “EMPLOYEE DETAILS” FROM employees;

SELECT employee\_id, last\_name, department\_id

FROM employees

WHERE LOWER(last\_name)=’higgins’;

|  |  |
| --- | --- |
| **Function** | **Result** |
| CONCAT(‘hello’, ‘world’) | helloworld |
| Substr(‘helloworld’,1,5) | Hello |
| Length(‘helloworld’) | 10 |
| Instr(‘helloworld’,’w’) | 6 |
| Lpad(salary,10,’\*’) | \*\*\*\*\*24000 |
| Rpad(salary,10,’\*’) | 24000\*\*\*\*\* |
| Trim(‘h’ FROM ‘helloworld’) | elloworld |

|  |  |  |
| --- | --- | --- |
| **Command** | **Query** | **Output** |
| initcap(char); | *select initcap(“hello”) from dual;* | Hello |
| lower (char);  upper (char); | *select lower (‘HELLO’) from dual;*  *select upper (‘hello’) from dual;* | Hello  HELLO |
| ltrim (char,[set]); | *select ltrim (‘cseit’, ‘cse’) from dual;* | IT |
| rtrim (char,[set]); | *select rtrim (‘cseit’, ‘it’) from dual;* | CSE |
| replace (char,search string, replace string); | *select replace (‘jack and jue’, ‘j’, ‘bl’) from dual;* | black and blue |
| substr (char,m,n); | *select substr (‘information’, 3, 4) from dual;* | form |

**Example:**

SELECT employee\_id, CONCAT (first\_name,last\_name) NAME , job\_id,LENGTH(last\_name), INSTR(last\_name,’a’) “contains’a’?”

FROM employees WHERE SUBSTR(job\_id,4)=’ERP’;

**NUMBER FUNCTIONS**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| round(column/expr, n) | Rounds the value to specified decimal |
| trunc(column/expr,n) | Truncates value to specified decimal |
| mod(m,n) | Returns remainder of division |

**Example**

|  |  |
| --- | --- |
| Function | Result |
| round(45.926,2) | 45.93 |
| trunc(45.926,2) | 45.92 |
| mod(1600,300) | 100 |

SELECT ROUND(45.923,2), ROUND(45.923,0), ROUND(45.923,-1) FROM dual;

**NOTE:**  Dual is a dummy table you can use to view results from functions and calculations.

SELECT TRUNC(45.923,2), TRUNC(45.923), TRUNC(45.923,-2) FROM dual;

SELECT last\_name,salary,MOD(salary,5000) FROM employees WHERE job\_id=’sa\_rep’;

**Working with Dates**

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.

– Enables you to store 21st-century dates in the 20th century by specifying only the last

two digits of the year

– Enables you to store 20th-century dates in the 21st century in the same way

**Example**

SELECT last\_name, hire\_date FROM employees WHERE hire\_date < '01-FEB-88;

**Working with Dates**

SYSDATE is a function that returns:

• Date

• Time

**Example**

**Display the current date using the DUAL table.**

SELECT SYSDATE FROM DUAL;

**Arithmetic with Dates**

• Add or subtract a number to or from a date for a resultant date value.

• Subtract two dates to find the number of days between those dates.

• Add hours to a date by dividing the number of hours by 24.

**Arithmetic with Dates**

Because the database stores dates as numbers, you can perform calculations using arithmetic

Operators such as addition and subtraction. You can add and subtract number constants as well as dates.

You can perform the following operations:

**Operation Result Description**

date + number Date Adds a number of days to a date

date – number Date Subtracts a number of days from a date

date – date Number of days Subtracts one date from another

date + number/24 Date Adds a number of hours to a date

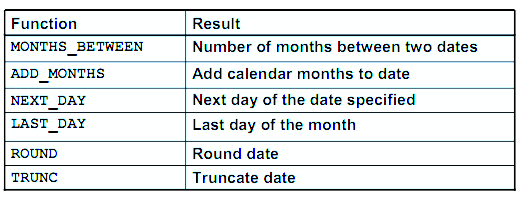
**Example**

SELECT last\_name, (SYSDATE-hire\_date)/7 AS WEEKS

FROM employees

WHERE department\_id = 90;

**Date Functions**



**Date Functions**

Date functions operate on Oracle dates. All date functions return a value of DATE data type

except MONTHS\_BETWEEN, which returns a numeric value.

• MONTHS\_BETWEEN(date1, date2)::: Finds the number of months between date1 and date2. The result can be positive or negative. If date1 is later than date2, the result is positive; if date1 is earlier than date2, the result is negative. The noninteger part of the result represents a portion of the month.

• ADD\_MONTHS(date, n)::: Adds n number of calendar months to date. The value of n must be an integer and can be negative.

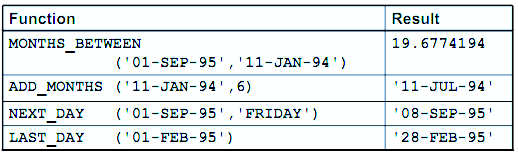
• NEXT\_DAY(date, 'char')::: Finds the date of the next specified day of the week ('char') following date. The value of char may be a number representing a day or a character string.

• LAST\_DAY(date)::: Finds the date of the last day of the month that contains date

• ROUND(date[,'fmt'])::: Returns date rounded to the unit that is specified by the format model fmt. If the format model fmt is omitted, date is rounded to the nearest day.

* TRUNC(date[, 'fmt'])::: Returns date with the time portion of the day truncated to the unit that is specified by the format model fmt. If the format model fmt is omitted, date is truncated to the nearest day.

**Using Date Functions**



**Example**

Display the employee number, hire date, number of months employed, sixmonth review date, first Friday after hire date, and last day of the hire month for all employees who have been employed for fewer than 70 months.

SELECT employee\_id, hire\_date,MONTHS\_BETWEEN (SYSDATE, hire\_date) TENURE,ADD\_MONTHS (hire\_date, 6) REVIEW,NEXT\_DAY (hire\_date, 'FRIDAY'), LAST\_DAY(hire\_date)

FROM employees

WHERE MONTHS\_BETWEEN (SYSDATE, hire\_date) < 70;

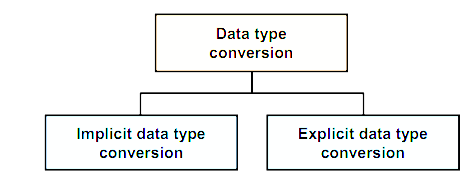
**Conversion Functions**

This covers the following topics:

• Writing a query that displays the current date

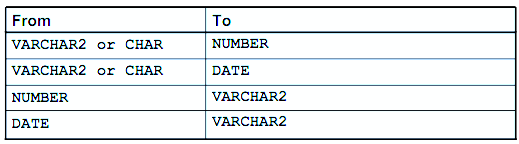
• Creating queries that require the use of numeric, character, and date functions

• Performing calculations of years and months of service for an employee



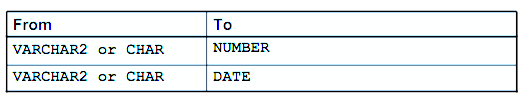
**Implicit Data Type Conversion**

For assignments, the Oracle server can automatically convert the following:

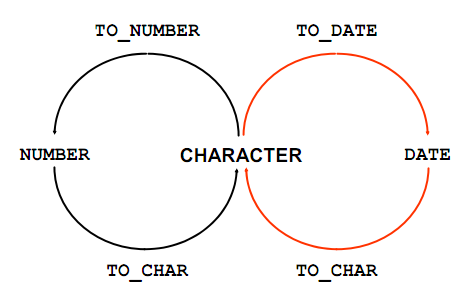


For example, the expression hire\_date > '01-JAN-90' results in the implicit conversion from the string '01-JAN-90' to a date.

For expression evaluation, the Oracle Server can automatically convert the following:



**Explicit Data Type Conversion**



**SQL provides three functions to convert a value from one data type to another:**

**Example:**

**Using the TO\_CHAR Function with Dates**

TO\_CHAR(date, 'format\_model')

**The format model:**

• Must be enclosed by single quotation marks

• Is case-sensitive

• Can include any valid date format element

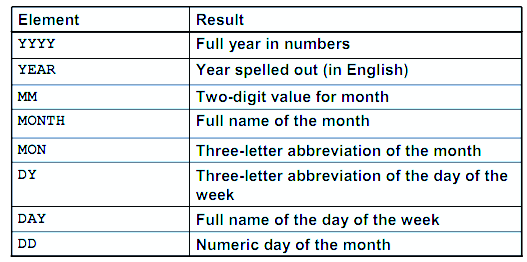
• Has an fm element to remove padded blanks or suppress leading zeros

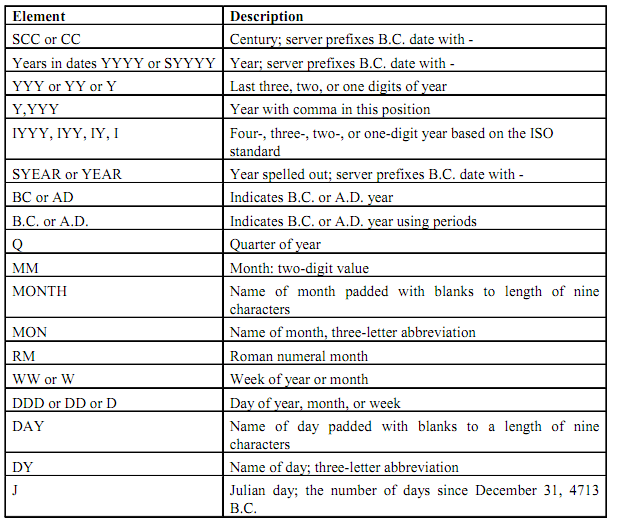
• Is separated from the date value by a comma

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

FROM employees WHERE last\_name = 'Higgins';

**Elements of the Date Format Model**

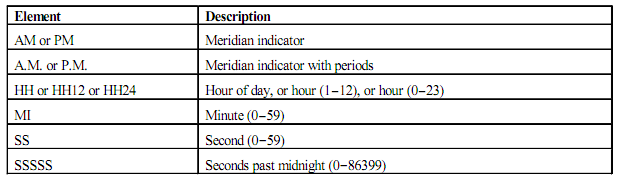


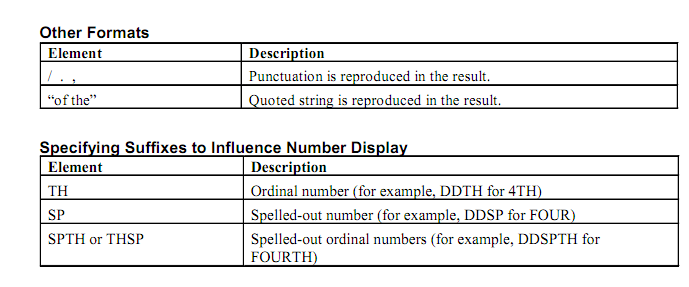
**Sample Format Elements of Valid Date** 

**Date Format Elements:** Time Formats

Use the formats that are listed in the following tables to display time information and literals

and to change numerals to spelled numbers.





**Example**

SELECT last\_name,

TO\_CHAR(hire\_date, 'fmDD Month YYYY') AS HIREDATE

FROM employees;

Modify example to display the dates in a format that appears as “Seventeenth of June 1987 12:00:00 AM.”

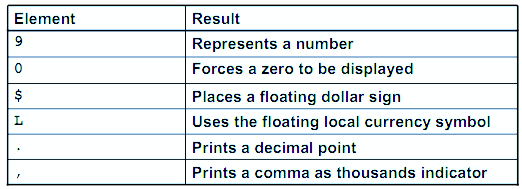
SELECT last\_name,

TO\_CHAR (hire\_date, 'fmDdspth "of" Month YYYY fmHH:MI:SS AM') HIREDATE

FROM employees;

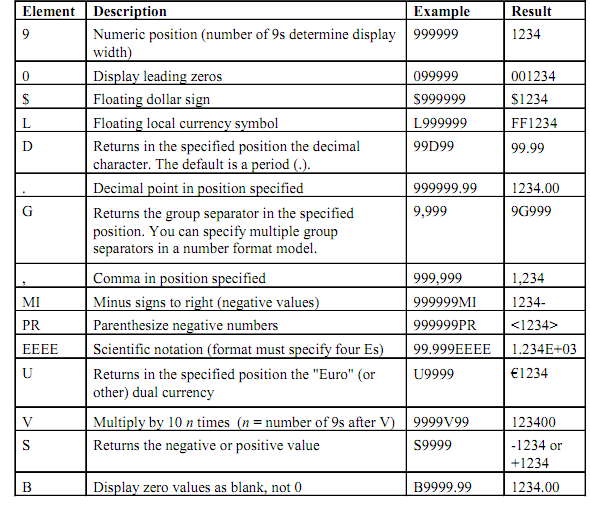
**Using the TO\_CHAR Function with Numbers**

TO\_CHAR(number, 'format\_model')  
These are some of the format elements that you can use with the TO\_CHAR function to display a number value as a character:



**Number Format Elements**

If you are converting a number to the character data type, you can use the following format elements:



SELECT TO\_CHAR(salary, '$99,999.00') SALARY

FROM employees

WHERE last\_name = 'Ernst';

**Using the TO\_NUMBER and TO\_DATE Functions**

• Convert a character string to a number format using the TO\_NUMBER function:

TO\_NUMBER(char[, 'format\_model']

• Convert a character string to a date format using the TO\_DATE function:

TO\_DATE(char[, 'format\_model']

• These functions have an fx modifier. This modifier specifies the exact matching for the character

argument and date format model of a TO\_DATE function.

The fx modifier specifies exact matching for the character argument and date format model of a TO\_DATE function:

• Punctuation and quoted text in the character argument must exactly match (except for case) the corresponding parts of the format model.

• The character argument cannot have extra blanks. Without fx, Oracle ignores extra blanks.

• Numeric data in the character argument must have the same number of digits as the corresponding element in the format model. Without fx, numbers in the character argument can omit leading zeros.

SELECT last\_name, hire\_date

FROM employees

WHERE hire\_date = TO\_DATE('May 24, 1999', 'fxMonth DD, YYYY');

**Questions**

**Q1)** Write a query to display the current date. Label the column Date.

# SQL>

**Q2)**The HR department needs a report to display the employee number, last name, salary, and increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

# SQL>

**Q3)**Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results by the employees’ last names.

# SQL>

**Q4)**Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

# SQL>

**Q5)**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

# SQL>

**Q6)**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.

# SQL>

**Q7)**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.

# SQL>

**Q8)**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.

# SQL>