SQL Functions

Chapter 10

DUMMY Table and Host Variables

- Both Oracle and DB2 support DUMMY Table and Host Variables

Oracle - DUAL Table

- Has one row called "X" and one column called "DUMMY"

DUMMY
X

- Used to create SELECT statements and execute functions not directly related to a specific database table
- Queries using the DUAL table return one row as a result

Oracle DUAL Table

```
SELECT (319/29) + 12
FROM DUAL;
```

(319/29)+12
23

```
SELECT (319/29) + 12 AS "Total"
FROM DUAL;
```

Total
23

DB2 - SYSIBM.SYSDUMMY1 and VALUES Keyword

- SYSIBM.SYSDUMMY1
  - A special in-memory table
- VALUES keyword

- Result set contains one row and one column
- Can be used with SELECT statements not accessing a database table directly

DB2

```
SELECT CURRENT_DATE
FROM SYSIBM.SYSDUMMY1;
```

```
00001
-----------
2020-03-09
```

```
VALUES CURRENT_DATE;
```

```
00001
-----------
2020-03-09
```

DB2

```
SELECT (319/29) + 12
FROM SYSIBM.SYSDUMMY1;
```

```
00001
-----
   23
```

```
VALUES (319/29) + 12;
```

```
00001
-----
   23
```

DB2 – Alias

```
SELECT (319/29) + 12 AS "Total"
FROM SYSIBM.SYSDUMMY1;
```

```
Total
-----
   23
```

- Cannot use an alias with the VALUES keyword

Host Variables – Page 235

- Called Bind variables in Oracle world
- Replace the hardcoded value in your statement with a :named_variable
- Starts with colon (:)
- DBMS will prompt for a value when statement is executed
- Alpha and date host variables are treated as character strings and require single quotation marks

Host Variables

- Original query:

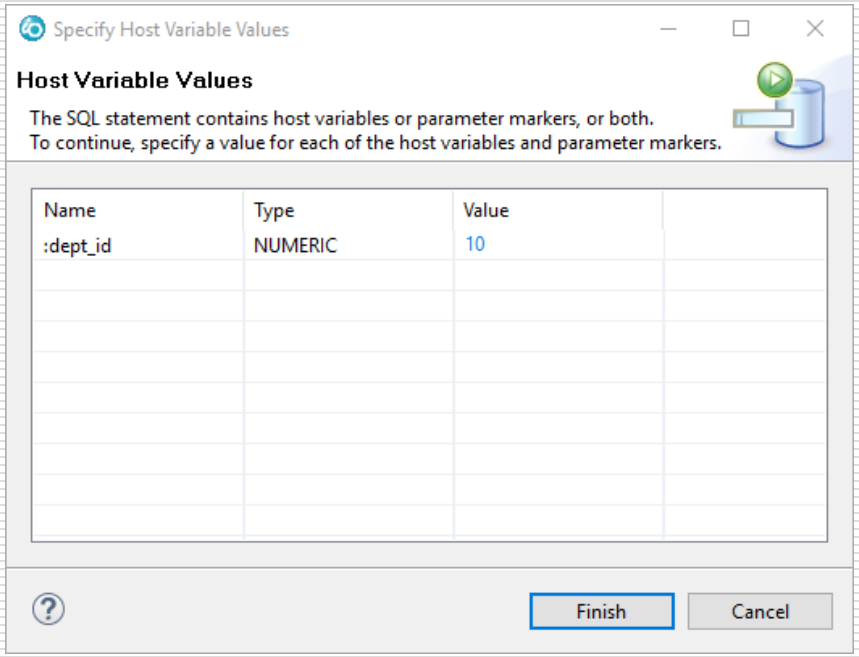
```
SELECT first_name, last_name, salary, department_id
  FROM employees
  WHERE department_id = 10; (and then 90, 110 . . .)
```

- With host variable

```
SELECT first_name, last_name, salary, department_id
  FROM employees
  WHERE department_id = :dept_id;
```

Host Variables

```
SELECT first_name, last_name, salary, department_id
  FROM employees
  WHERE department_id = :dept_id;
```




```
FIRST_NAME LAST_NAME SALARY   DEPARTMENT_ID
---------- --------- -------  -------------
Jennifer   Whalen    4400.00            10
```

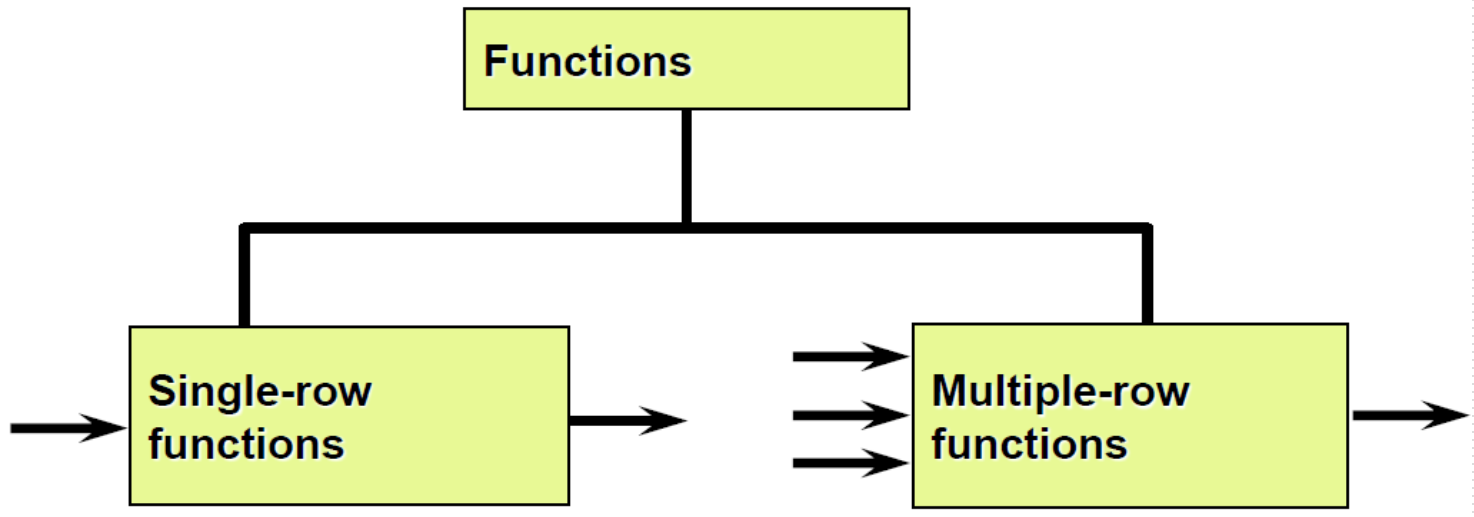
SQL Functions

SQL Functions

- Used to manipulate data values
- Predefined block of code that accepts arguments (parameters) and returns output, usually different from the input value
- Functions have both input and output

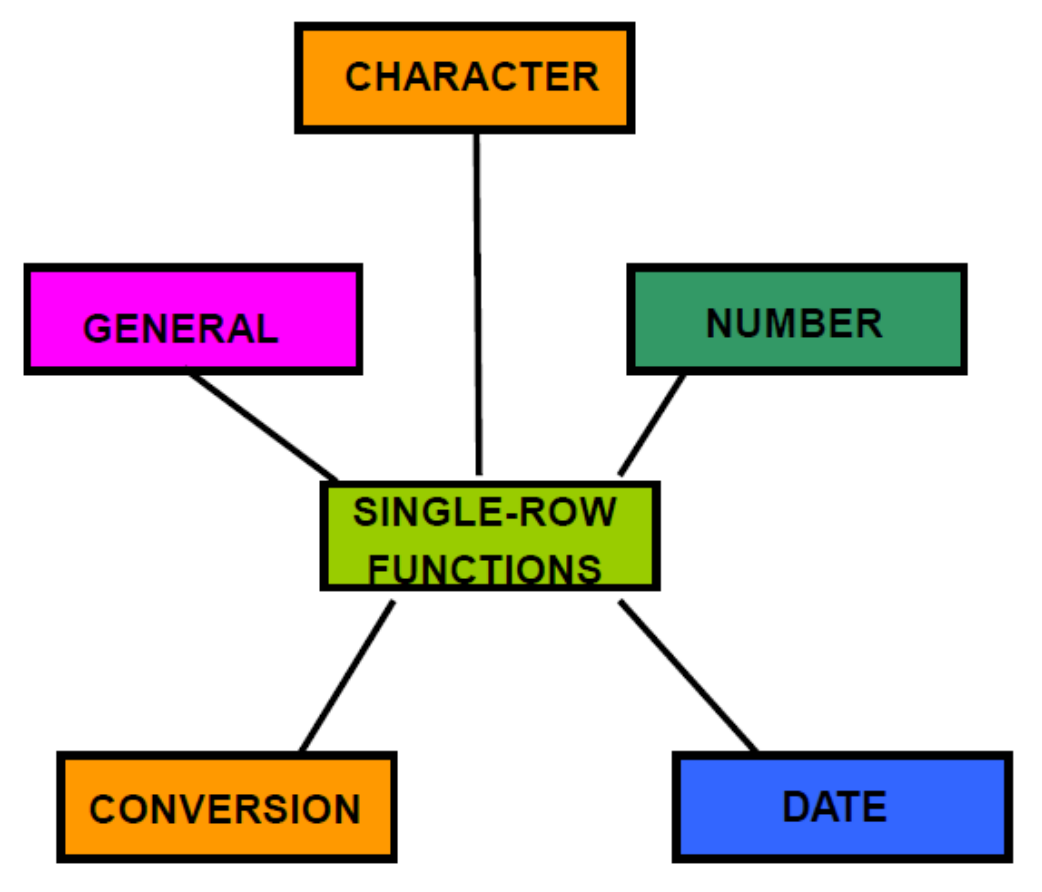
Types of Functions

- Two distinct types of functions:
  - Single-row functions – manipulates one row and returns one result
  - Multiple-row functions – manipulates a group of rows and returns one result per group of rows



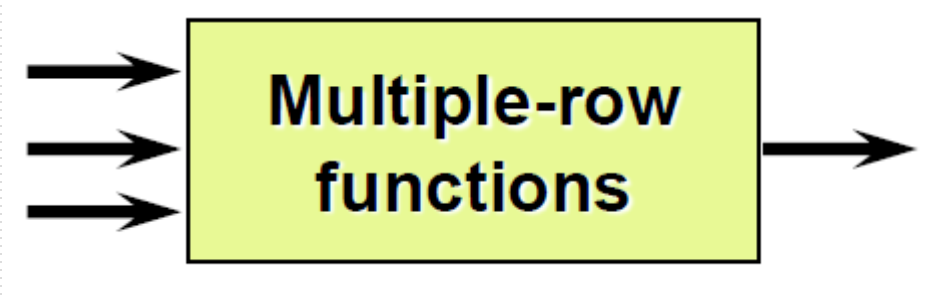
Single-Row Functions

- Accepts one or more arguments and returns a single result
- If a single-row function is applied to 12 rows, 12 results are returned from the single-row function



Multiple-Row (Group) Functions

- Accepts many rows as input and returns a single value as output
- Known as group functions
- The input rows may be all rows in a table or a subset of the rows in the table



Multiple-Row Functions

- Operate on sets of rows to give one result per group

Function	Returns
AVG	The average value of a given column
COUNT	The total number of values in a given column
COUNT(*)	The total number of rows in a table
MAX	The largest value in a given column
MIN	The smallest value in a given column
SUM	The sum of the numeric values in a given column

COUNT(*) & COUNT(column_name)

- COUNT(*) – Returns count of all rows in the table
- COUNT(column_name) – Returns count of rows that do not have NULL values for the specified column

```
SELECT COUNT(*) AS total_employees
  FROM employees;
Results:
TOTAL_EMPLOYEES
40
```

```
SELECT COUNT(commission_pct) AS employees_assigned_pct
  FROM employees;
Results:
EMPLOYEES_ASSIGNED_PCT
9
```

COUNT & DISTINCT

```
SELECT job_id
  FROM job_history;
Results:
JOB_ID
AC_ACCOUNT
AC_ACCOUNT
AC_MGR
AD_ASST
IT_PROG
MK_REP
SA_MAN
SA_REP
ST_CLERK
ST_CLERK
```

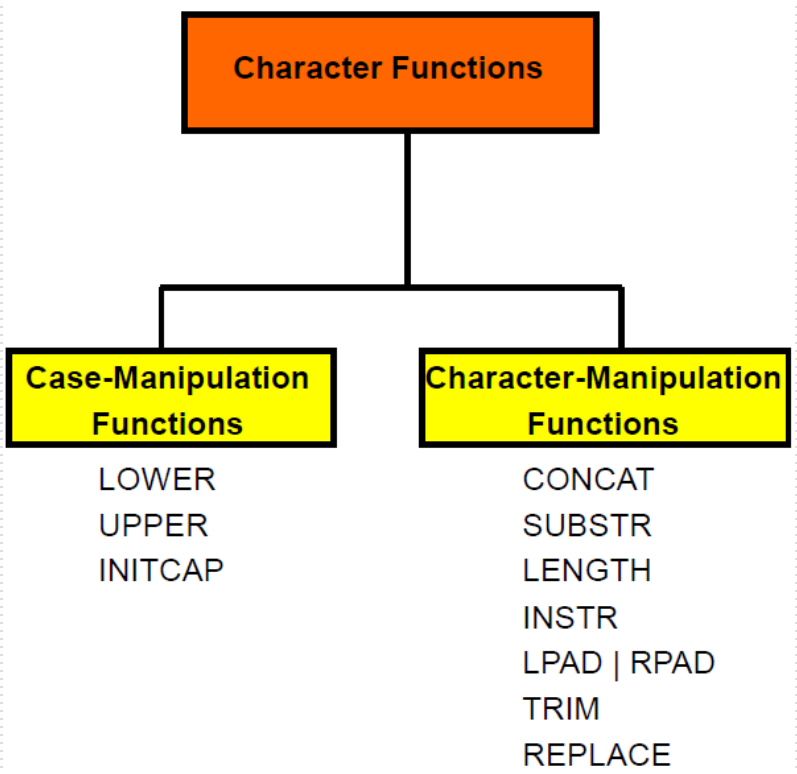
```
SELECT COUNT(job_id)
  FROM job_history;
Results:
COUNT(JOB_ID)
10
```

```
SELECT COUNT(DISTINCT job_id)
  FROM job_history;
Results:
COUNT(DISTINCT JOB_ID)
8
```

Single-Row Character Functions

Single-Row Character Functions

- Two categories:
  - Case-Manipulation Functions
    - Convert the case of character strings
  - Character-Manipulation Functions
    - Manipulate character strings - join, extract, show, find, pad, and trim



- Can be used in the SELECT, WHERE, and ORDER BY clauses

Case Manipulation Functions

- LOWER
- UPPER

LOWER Function

- Converts alpha characters to lowercase letters
- When used in a SELECT clause, the appearance of the data in the results set is altered

```
SELECT first_name, last_name, job_id, LOWER(job_id) AS job_id
  FROM employees
  WHERE job_id = 'SA_REP';
```

FIRST_NAME	LAST_NAME	JOB_ID	JOB_ID
Ellen	Abel	SA_REP	sa_rep
Jonathon	Taylor	SA_REP	sa_rep
Kimberely	Grant	SA_REP	sa_rep

LOWER Function

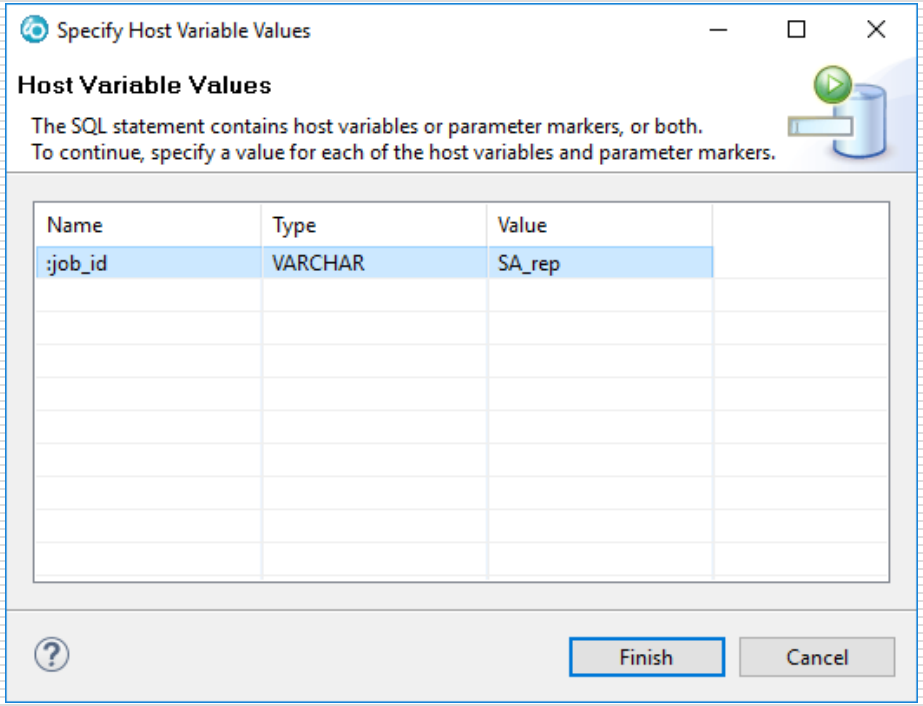
- When used in a WHERE clause, the value in the comparison is altered

```
SELECT first_name, last_name, job_id
  FROM employees
  WHERE LOWER(job_id) = 'sa_rep';
```

FIRST_NAME	LAST_NAME	JOB_ID
Ellen	Abel	SA_REP
Jonathon	Taylor	SA_REP
Kimberely	Grant	SA_REP

LOWER Function

```
SELECT first_name, last_name, job_id
  FROM employees
  WHERE LOWER(job_id) = LOWER(:job_id);
```




```
FIRST_NAME LAST_NAME JOB_ID
---------- --------- ------
Ellen      Abel      SA_REP
Jonathon   Taylor    SA_REP
Kimberely  Grant     SA_REP
```

UPPER Function

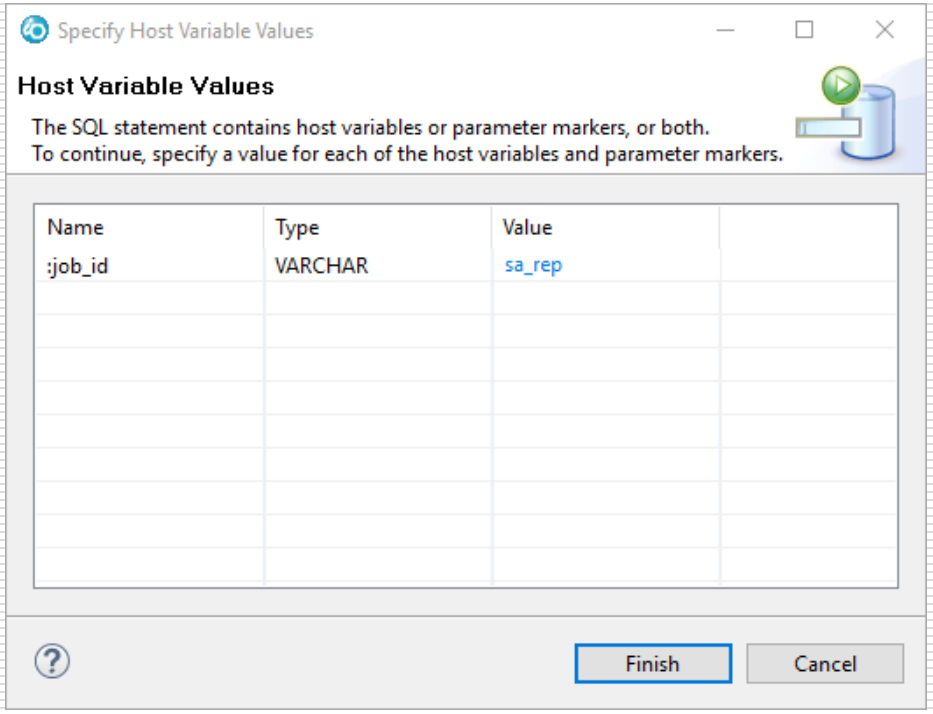
- Convert alpha characters to uppercase letters

```
SELECT first_name, last_name, job_id
  FROM employees
  WHERE job_id = UPPER('sa_rep');
```

FIRST_NAME	LAST_NAME	JOB_ID
Ellen	Abel	SA_REP
Jonathon	Taylor	SA_REP
Kimberely	Grant	SA_REP

UPPER Function

```
SELECT first_name, last_name, job_id
  FROM employees
  WHERE UPPER(job_id) = UPPER(:job_id);
```




```
FIRST_NAME LAST_NAME JOB_ID
---------- --------- ------
Ellen      Abel      SA_REP
Jonathon   Taylor    SA_REP
Kimberely  Grant     SA_REP
```

Character Manipulation Functions

- Used to extract, change, format, or alter a character string
- Character Manipulation Functions:
  - Accept one or more characters or words as arguments (parameters)
  - Perform its functionality on the input character strings
  - Return the changed, extracted, counted, or altered value

CONCAT Function

- || (pipes) – ANSI Standard

- CONCAT Function:
  - Joins two values together
    - Accepts two character strings as arguments
    - Joins the strings
    - Outputs one string

|| and CONCAT Functions

```
SELECT first_name, last_name, city || ', ' || state || ' ' || zip AS address
  FROM f_customers;

SELECT first_name, last_name,
       city CONCAT ', ' CONCAT state CONCAT ' ' CONCAT zip AS address
  FROM f_customers;
```

```
FIRST_NAME LAST_NAME ADDRESS
---------- --------- -----------------
Cole       Bee       Orlando, FL 32838
Zoe        Twee      Boston, MA 12889
```

CONCAT Function

```
SELECT CONCAT('Customer# ', id) AS customer_no,
       first_name || ' ' || last_name  AS customer_name
  FROM f_customers;
```

```
CUSTOMER_NO   CUSTOMER_NAME
------------- -------------
Customer# 123 Cole Bee
Customer# 456 Zoe Twee
```

SUBSTR Function

```
substr(string, position)
substr(string, position, length)
```

- SUBSTR:
  - Accepts three arguments (character string, starting position, length)
    - Extracts a substring from the character string argument
    - Starting at the position specified by the starting position argument
    - Returns a string for a length specified by the length argument (optional)

SUBSTR Function

```
substr(string, position, length)
```

- Position parameter:
  - Starting position
  - If the position parameter is 0, it defaults to 1
- Length parameter
  - Length of string to be extracted
  - Optional, and if omitted, returns all characters to the end of the string

SUBSTR Function

substr(string, position)

```
VALUES SUBSTR('1234567890',4);
```

4567890

substr(string, position, length)

```
VALUES SUBSTR('1234567890',4, 3);
```

456

SUBSTR Function

- Position parameter can be negative, in which case it is counted from the right side:

```
SELECT SUBSTR('1234567890',-4, 3) FROM SYSIBM.SYSDUMMY1;
```

789

LENGTH Function

- Accepts a character string as an argument and returns the number of characters (length) in that character string

```
SELECT department_name, LENGTH(department_name) AS length
FROM departments;
```

```
DEPARTMENT_NAME LENGTH
--------------- ------
Administration      14
Marketing            9
Shipping             8
IT                   2
Sales                5
Executive            9
Accounting          10
Contracting         11
```

INSTR (In String) Function

- Searches for the first occurrence of a substring within a character string and returns the position as a number

- If the substring is not found, zero is returned

INSTR Function

```
INSTR( source_string, substring [, start_position [, occurrance ] ] )
```

- source_string - the string to be searched
- substring - the character string to be searched for inside of source_string
- start_position - an optional argument. It is an integer value that tells where to start searching in the source_string. If the start_position is negative, then it counts back that number of characters from the end of the source_string and then searches backwards from that position. If omitted, this defaults to 1
- occurrence - an integer indicating which occurrence of substring should be search for. That is, should INSTR return the first matching substring, the second matching substring, etc. This argument is optional. If omitted, it defaults to 1
- If the substring is not found in source_string, the INSTR function returns 0.

INSTR Function

INSTR(*source_string*, *substring* [, ***start_position*** [, *occurrance*]])

```
VALUES INSTR('abcdabcd','b');      (returns 2)
VALUES INSTR('bacdabcb','b',1);    (returns 1)
VALUES INSTR('abcdabcd','b',1);    (returns 2)
VALUES INSTR('abcdabcd','b',4);    (returns 6)
VALUES INSTR('abcdabcd','b',0);    (returns 0 or NULL)
VALUES INSTR('abcdabcd','b',-1);   (returns 6)
VALUES INSTR('abbdabcd','b',-4);   (returns 3)
VALUES INSTR('abcdabcd','z');      (returns 0) – substring not found
```

- If start_position is 0, 0 or NULL is returned

INSTR Function

INSTR(*source_string*, *substring* [, ***start_position*** [, ***occurrance***]])

```
VALUES INSTR('abcdabcd','b',1,1); (returns 2)
VALUES INSTR('abcdabcd','b',1,2); (returns 6)
VALUES INSTR('abcdabcd','b',1,0);
   Oracle - (returns: ORA-01428: argument '0' is out of range)
   DB2 – returns NULL
VALUES INSTR('abcdabcd','b',1,-1);
   Oracle - (returns: ORA-01428: argument '-1' is out of range)
   DB2 – returns NULL
VALUES INSTR('abcdabcd','b',-1,1); (returns 6)
VALUES INSTR('abcbbbcd','b',-1,2); (returns 5)
VALUES INSTR('abcdabcb','b',-1,1); (returns 8)
```

LPAD Function

- Pads (adds characters to) the left-side of a character string, resulting in a right-justified value

LPAD Function

```
LPAD(string, padding_length, [ padding_string ])
```

The *string* that is being modified. The padded characters are added to the left-side of the string

The *padding_length* is the number of characters to *return* (**not** the number of characters to add). If the *padding_length* is smaller than the original string, the LPAD function will truncate the string to the size of the *padding_length*.

The *padding_string* is optional. This is the character string that will be padded to the left-hand side of *string*. If this parameter is omitted, the LPAD function will default to padding space characters to the left-side of the *string*.

LPAD Function

```
LPAD(string, padding_length, [ padding_string ])

VALUES LPAD('Canada', 10);          returns '    Canada'

VALUES LPAD('Canada', 2);           returns 'Ca'

VALUES LPAD('Canada', 10, 'X');     returns 'XXXXCanada'

VALUES LPAD('Canada', 6, 'X');      returns 'Canada'
```

LPAD Function

```
SELECT salary, LPAD(salary,12,'$'), LPAD(salary,12,'*')
  FROM employees
  WHERE salary > 15000;
```

```
SALARY   00002        00003
-------- ------------ ------------
24000.00 $$$$24000.00 ****24000.00
17000.00 $$$$17000.00 ****17000.00
17000.00 $$$$17000.00 ****17000.00
```

RPAD Function

- Pads the right-side of a character string, resulting in a left-justified value

RPAD (input_string, length, padding_character)

input-string being modified. The padded characters are added to the left-side of the string

length is the net length of the string including padding

padding_character is the character to be used for padding

RPAD Function

```
SELECT first_name,
       RPAD(first_name, 10, ' '),
       RPAD(first_name, 10, '-')
  FROM employees;
```

```
FIRST_NAME 00002      00003
---------- ---------- ----------
Steven     Steven     Steven----
Neena      Neena      Neena-----
Lex        Lex        Lex-------
Jennifer   Jennifer   Jennifer--
Shelley    Shelley    Shelley---
William    William    William---
Eleni      Eleni      Eleni-----
Ellen      Ellen      Ellen-----
Jonathon   Jonathon   Jonathon--
Kimberely  Kimberely  Kimberely-
```

TRIM Functions

- Three types:
  - LTRIM – Trim left
  - RTRIM – Trim right
  - TRIM – Trim both

LTRIM Function

- LTRIM(string1 [, trim_string])

- Remove a specific string of characters from the left side of a string or column
- Default is blank
- Returns a value with data type VARCHAR

LTRIM Function

LTRIM(string1 [, ~~trim_string~~])

```
SELECT 'Lambton' || ' ' || LTRIM('     College')
FROM SYSIBM.SYSDUMMY1;

VALUES 'Lambton' || ' ' || LTRIM('     College');
```

```
00001
----------------
Lambton College
```

LTRIM Function

```
LTRIM( string1 [, trim_string] )
```

```
SELECT address, LTRIM(address, 'P.O. ')
FROM customers
WHERE state = 'CA';
```

ADDRESS	LTRIM(ADDRESS,'P.O.')
P.O. BOX 9835	BOX 9835
P.O. BOX 8564	BOX 8564
9851231 LONG ROAD	9851231 LONG ROAD

RTRIM Function

```
RTRIM( string1 [, trim_string] )
```

- Remove a specific string of characters from the right side of a string
- Returns a value of VARCHAR data type

RTRIM Function

RTRIM(string1 [, ~~trim_string~~])

```
SELECT RTRIM('Lambton     ') || ' ' || LTRIM('College')
FROM SYSIBM.SYSDUMMY1;

VALUES RTRIM('Lambton     ') || ' ' || LTRIM('College');
```

```
00001
----------------
Lambton College
```

TRIM Function

- Removes all specified characters from either the beginning, the end, or both beginning and end of a string
- Optional parameters indicate whether leading, or trailing, or both leading and trailing pad characters should be removed
- Returns a value with data type VARCHAR

TRIM Function

```
VALUES TRIM('   Los   ') || ' ' || TRIM('   Angeles   ');
```

```
00001
-----------
Los Angeles
```

```
VALUES TRIM('*' FROM '***Los***') || ' ' || TRIM('*' FROM '***Angeles***');
```

```
00001
-----------
Los Angeles
```

```
VALUES TRIM(LEADING '*' FROM '***Los***') || ' ' || TRIM(LEADING '*' FROM '***Angeles***');
```

```
00001
-----------------
Los*** Angeles***
```

```
VALUES TRIM(TRAILING '*' FROM '***Los***') || ' ' || TRIM(TRAILING '*' FROM '***Angeles***');
```

```
00001
-----------------
***Los ***Angeles
```

TRIM LEADING/TRAILING Function

```
VALUES TRIM(BOTH '*' FROM '***Los***') || ' ' || TRIM(BOTH '*' FROM '***Angeles***');
```

```
00001
-----------
Los Angeles
```

```
VALUES TRIM(LEADING '*' FROM TRIM(TRAILING 'X' FROM '***CanadaXXX'));
```

```
00001
------
Canada
```

REPLACE Function

- The REPLACE function replaces all occurrences of a substring within a string with a new substring
- Changes one pattern with another

```
REPLACE (string1, string_to_replace, [replacement_string] )
```

- string1 is the string that will have characters replaced in it
- string_to_replace is the string that will be searched for and taken out of string1
- [replacement_string] is the new string to be inserted in string1

REPLACE Function

```
REPLACE (string1, string_to_replace, [replacement_string] )
```

```
VALUES REPLACE('JACK and JUE','J','BL');
```

```
00001
--------------
BLACK and BLUE
```

REPLACE Function

```
REPLACE (string1, string_to_replace, [replacement_string] )
```

```
SELECT address, REPLACE(address, 'P.O.', 'POST OFFICE')
  FROM customers
  WHERE state = 'CA';
```

ADDRESS	REPLACE(ADDRESS,'P.O.','POSTOFFICE')
P.O. BOX 9835	POST OFFICE BOX 9835
P.O. BOX 8564	POST OFFICE BOX 8564
9851231 LONG ROAD	9851231 LONG ROAD

REPLACE Function

REPLACE (string1, string_to_replace, [~~replacement_string~~])

- No replacement_string

```
SELECT last_name, job_id, REPLACE(job_id, 'SA_')
  FROM employees
  WHERE job_id LIKE 'SA%';
```

LAST_NAME	JOB_ID	REPLACE(JOB_ID,'SA_')
Zlotkey	SA_MAN	MAN
Abel	SA_REP	REP
Taylor	SA_REP	REP
Grant	SA_REP	REP

REPLACE Function

```
VALUES REPLACE ('ABCDEAB','AB','acvv');
```

Result: acvvCDEacvv

TRANSLATE

- Changes character by character
- If no replacement character is specified, the character is replaced with blank

```
TRANSLATE( string1, replacement_string, string_to_replace )

    VALUES TRANSLATE ('ABCDE', 'ac', 'AC');
    Result: aBcDE

    VALUES TRANSLATE ('ABCED', 'ac', 'ACB');
    Result: a cDE
```

TRANSLATE Function

```
SELECT phone_number,
       TRANSLATE(phone_number, '-', '.') AS new_phone
  FROM employees;
```

PHONE_NUMBER	NEW_PHONE
515.123.4567	515-123-4567
515.123.4568	515-123-4568
515.123.4569	515-123-4569
515.123.4444	515-123-4444
515.123.8080	515-123-8080
515.123.8181	515-123-8181
011.44.1344.429018	011-44-1344-429018
011.44.1644.429267	011-44-1644-429267
011.44.1644.429265	011-44-1644-429265
011.44.1644.429263	011-44-1644-429263

Column Aliases with Functions

```
SELECT SUBSTR(first_name,1,1) || '. ' || last_name AS "Employee Name"
FROM employees
WHERE department_id = 110;
```

```
Employee Name
-------------
S. Higgins
W. Gietz
```


Q & A
Questions &
Answers