GIFT School of Engineering and Applied Sciences

Fall 2022

CS-244: Database Systems-Lab

Lab-11 Manual

Single Row Functions in SQL

Introduction to Lab

Functions make the basic query block more powerful, and they are used to manipulate data values. This is the first of two labs that explore functions. It focuses on single-row character, number. The main topics of this lab include:

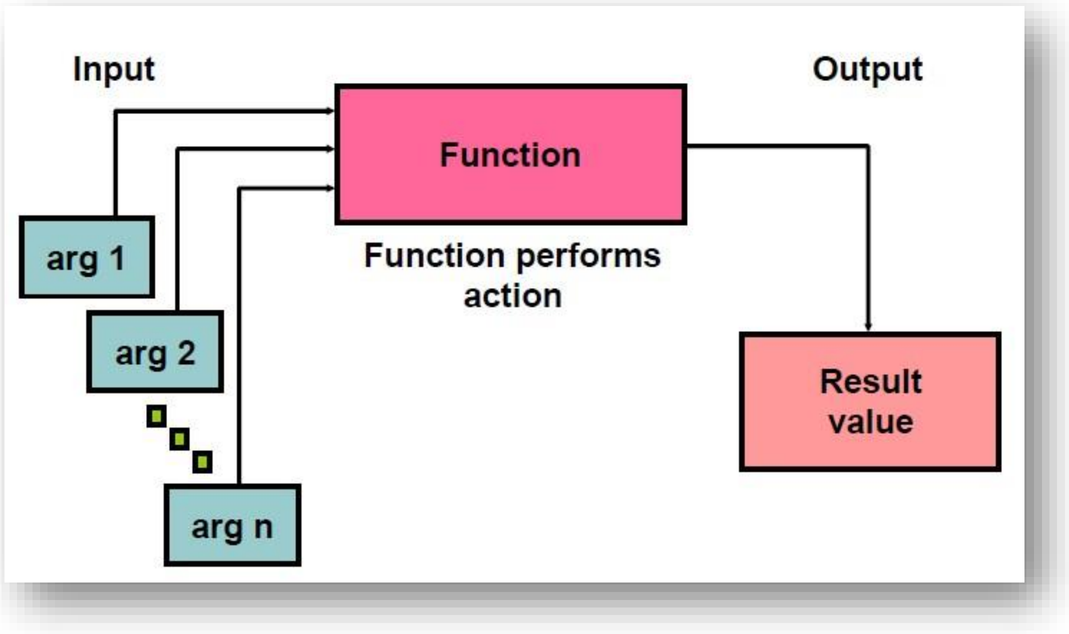
1. SQL Functions
2. Two Types of SQL Functions
3. Single-Row Functions
4. Character Functions
5. Case-Manipulation Functions
6. Using Case-Manipulation Functions
7. Character-Manipulation Functions
8. Using the Character-Manipulation Functions
9. Number Functions
10. Using the ROUND Function
11. Using the TRUNC Function
12. Using the MOD Function
13. NVL Function
14. Practice SQL Statements
15. SOLUTIONS: Practice SQL Statements

Objectives of this Lab

At the end of this lab, students should be able to:

1. Describe various types of functions that are available in SQL
2. Use character, number in SELECT statements
3. Understand the use of NVL function in case of NULL values

1. SQL Functions



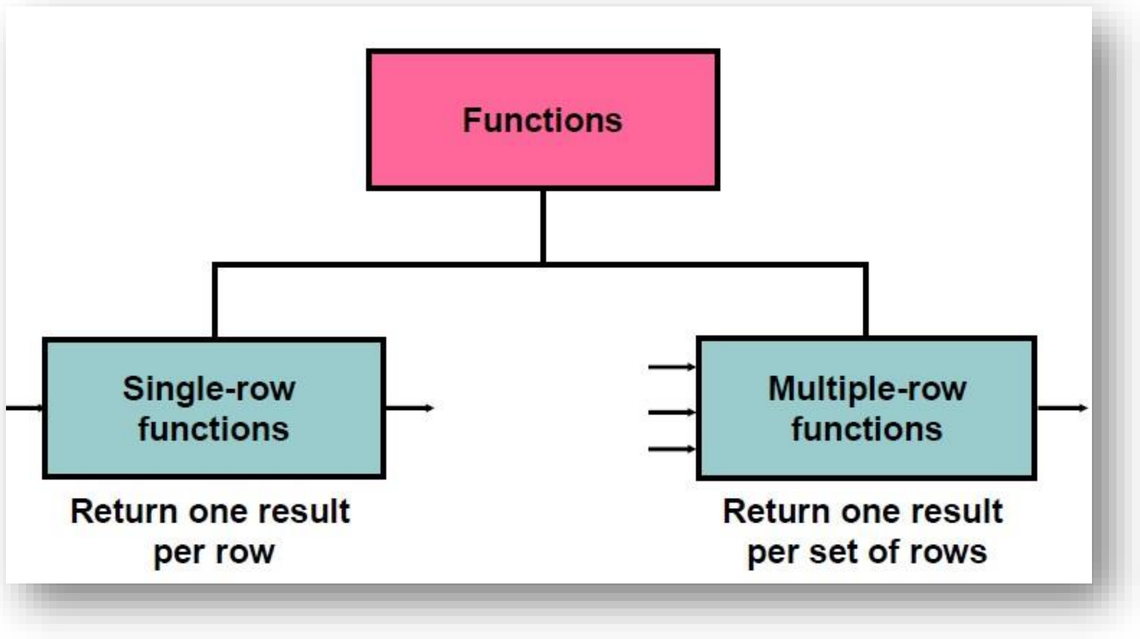
Functions are a very powerful feature of SQL. They can be used to do the following:

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

SQL functions sometimes take arguments and always return a value.

Note: Most of the functions that are described in this lab are specific to the Oracle version of SQL.

2. Two Types of SQL Functions



There are two types of functions:

- Single-row functions
- Multiple-row functions

Single-Row Functions

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

Multiple-Row Functions

Functions can manipulate groups of rows to give one result per group of rows. These functions are also known as *group functions* (covered in a later lab).

3. Single-Row Functions

```
function_name [(arg1, arg2,...)]
```

Single-row functions are used to manipulate data items. They accept one or more arguments and return one value for each row that is returned by the query. An argument can be one of the following:

- User-supplied constant

- Variable value
- Column name
- Expression

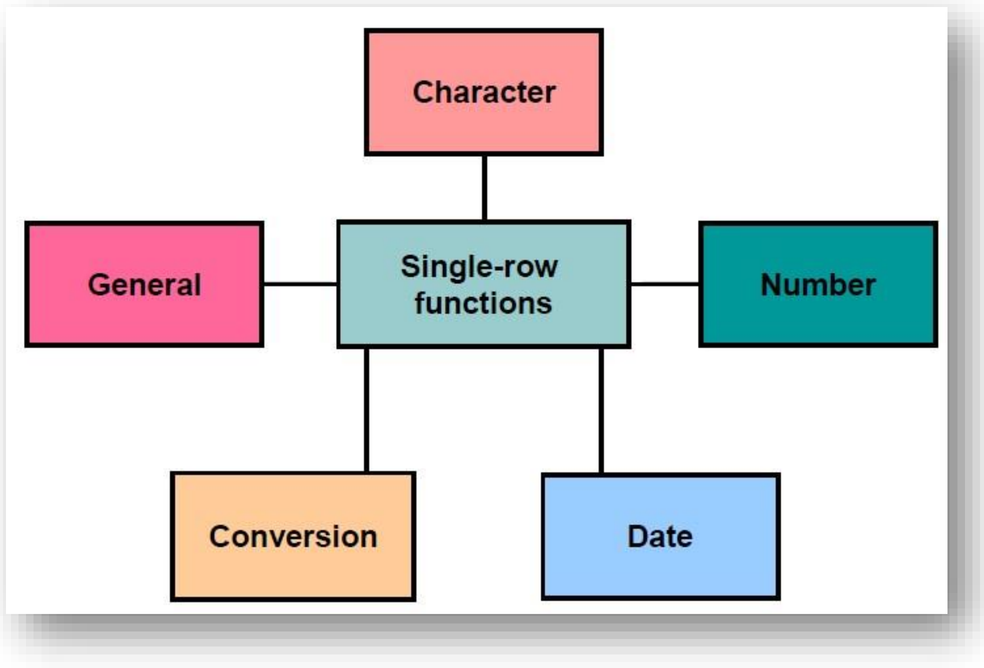
Features of single-row functions include:

- Acting on each row that is returned in the query
- Returning one result per row
- Possibly returning a data value of a different type than the one that is referenced
- Possibly expecting one or more arguments
- Can be used in **SELECT**, **WHERE**, and **ORDER BY** clauses; can be nested

In the syntax: ***function_name*** is the name of the

function

arg1, arg2 is any argument to be used by the function. This can be represented by a column name or expression.



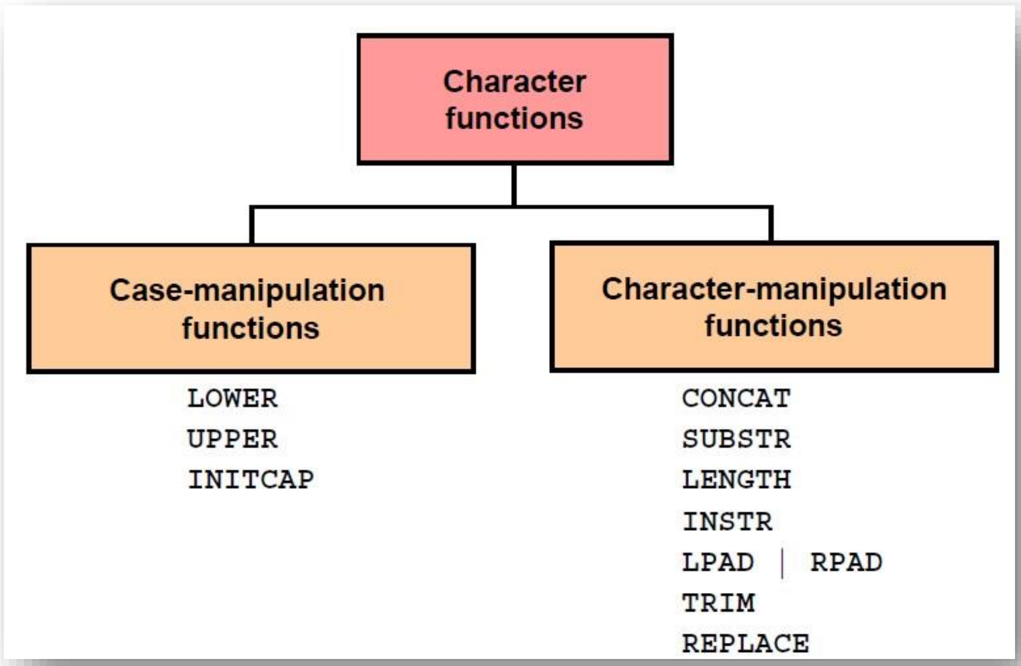
This lab covers the following single-row functions:

- **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**

4. Character Functions



Single-row character functions accept character data as input and can return both character and numeric values. Character functions can be divided into the following:

- Case-manipulation functions
- Character-manipulation functions

5. Case-Manipulation Functions

LOWER, **UPPER**, and **INITCAP** are the three case-conversion functions.

- **LOWER**

Converts mixed-case or uppercase character strings to lowercase

- **UPPER**
  Converts mixed-case or lowercase character strings to uppercase
- **INITCAP**
  Converts the first letter of each word to uppercase and remaining letters to lowercase

```
SELECT 'The job id for ' || UPPER(ename)|| ' is ' || LOWER(job)
AS "EMPLOYEE DETAILS" FROM emp;
```

6. Using Case-Manipulation Functions

Display the employee number, name, and department number for employee *Clark*:

```
SELECT empno, ename, deptno
FROM emp WHERE ename = 'clark';
 no rows selected
```

As the employee names are stored as capital strings, no rows are selected using the above query.

However, the correct version of the above query may be written as:

```
SELECT empno, ename, deptno
FROM emp
WHERE LOWER(ename) = 'clark';
```

7. Character-Manipulation Functions

CONCAT, **SUBSTR**, **LENGTH**, **INSTR**, **LPAD**, **RPAD**, and **TRIM** are the character manipulation functions that are covered in this lab.

- **CONCAT**
  - Joins values together (You are limited to using two parameters with **CONCAT**.)
- **SUBSTR**
  - Extracts a string of determined length
- **LENGTH**
  - Shows the length of a string as a numeric value
- **INSTR**
  - Finds the numeric position of a named character
- **LPAD**
  - Pads the character value right-justified
- **RPAD**
  - Pads the character value left-justified
- **TRIM**
  - Trims heading or trailing characters (or both) from a character string (If ***trim_character*** or ***trim_source*** is a character literal; you must enclose it in single quotation marks.)

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*****`
`REPLACE ('JACK and JUE','J','BL')`	`BLACK and BLUE`
`TRIM('H' FROM 'HelloWorld')`	`elloWorld`

8. Using the Character-Manipulation Functions

This example displays employee names and employee numbers joined together, the length of the employee name, and the numeric position of the letter *a* in the employee name for all employees who have the string **MAN** contained in the job starting at the sixth position of the job.

```
SELECT empno, CONCAT(empno, ename) NAME, job,
     LENGTH (ename), INSTR(ename, 'a') "Contains
     'a'?" FROM emp WHERE SUBSTR(job, 6) =
     'MAN';
```

Modify the SQL statement in the example to display the data for those employees whose names end with the letter *n*.

```
SELECT empno, CONCAT(empno, ename) NAME,

LENGTH (ename), INSTR(ename, 'a') "Contains 'a'?"

FROM emp

WHERE SUBSTR(ename, -1, 1) = 'n';
```

9. Number Functions

Number functions accept numeric input and return numeric values. This section describes some of the number functions.

Function	Purpose
ROUND(*column*|*expression*, *n*)	Rounds the column, expression, or value to *n* decimal places or, if *n* is omitted, no decimal places (If *n* is negative, numbers to left of the decimal point are rounded.)
TRUNC(*column*|*expression*, *n*)	Truncates the column, expression, or value to *n* decimal places or, if *n* is omitted, *n* defaults to zero
MOD(*m*,*n*)	Returns the remainder of *m* divided by *n*

10. Using the ROUND Function

The **ROUND** function rounds the column, expression, or value to *n* decimal places. If the second argument is 0 or is missing, the value is rounded to zero decimal places. If the second argument is 2, the value is rounded to two decimal places. Conversely, if the second argument is –2, the value is rounded to two decimal places to the left (rounded to the nearest unit of 10).

The **ROUND** function can also be used with date functions. You will see examples later in this lab.

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

11. Using the TRUNC Function

The **TRUNC** function truncates the column, expression, or value to *n* decimal places.

The **TRUNC** function works with arguments similar to those of the **ROUND** function. If the second argument is 0 or is missing, the value is truncated to zero decimal places. If the second argument is 2, the value is truncated to two decimal places. Conversely, if the second argument is –2, the value is truncated to two decimal places to the left. If the second argument is –1, the value is truncated to one decimal place to the left.

Like the **ROUND** function, the **TRUNC** function can be used with date functions.

```
SELECT TRUNC(45.923,2), TRUNC (45.923),
TRUNC (45.923,-1) FROM DUAL;
```

12. Using the MOD Function

The **MOD** function finds the remainder of the first argument divided by the second argument.

This example calculates the remainder of the salary after dividing it by 500 for all employees whose job is SALESMAN.

```
SELECT ename, sal, MOD(sal, 500)
FROM emp
WHERE job = 'SALESMAN';
```

Note: The **MOD** function is often used to determine if a value is odd or even

13. NVL Function

To convert a null value to an actual value, use the NVL function.

Syntax

```
NVL (expr1, expr2)
```

In the syntax:

- ***expr1*** is the source value or expression that may contain a null
- ***expr2*** is the target value for converting the null

You can use the **NVL** function to convert any data type, but the return value is always the same as the data type of ***expr1***.

Data types that can be used are date, character, and number. Data types must match:

- `NVL(comm,0)`
- `NVL(hiredate,'01-JAN-97')`
- `NVL(job,'No Job Yet')`

Example:

To calculate the annual compensation of all employees, you need to multiply the monthly salary by 12 and then add the commission percentage to the result:

```
SELECT ename, sal, NVL(comm, 0), (sal*12) + (sal*12*NVL(comm, 0))
AN_SAL FROM emp;
```

14. Practice SELECT Statements

Write SELECT statements for the following:

1. Write a query to display the current date. Label the column **Date.**
2. The HR department needs a report to display the employee number, name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column *New Salary*.
3. Modify above query to add a column that subtracts the old salary from the new salary. Label the column Increase.
4. Write a query that displays the employee 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' names.

5. Create a report that produces the following for each employee: <employee name> earns <salary> monthly but wants <3 times salary>. Label the column *Dream Salaries*.
6. 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*.
7. Create a query that displays the first eight characters of the employees' names and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column *EMPLOYEES_AND_THEIR_SALARIES*.
8. The HR department wants to find the length of employment for each employee. For each employee, display the 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.
9. 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*.

15. SOLUTIONS: Practice SELECT Statements

The End