**Lab 03: Single-row and multiple-row functions in SQL**

**What Are SQL Functions?**

SQL functions are special tools in SQL (Structured Query Language) that help you do different tasks with your data. They can:

* **Calculate numbers** (like adding or averaging).
* **Change individual pieces of data** (like updating a name).
* **Work with groups of rows** (like getting the total sales for a month).
* **Format dates and numbers** (like changing how a date looks).
* **Change the type of data in a column** (like turning a number into text).

SQL functions take **inputs** (called arguments) and always give you back a **result**.

**Types of SQL Functions**

There are **two main types** of SQL functions:

1. **Single-Row Functions**
   * These functions work on one row of data at a time and give you one result for each row.
   * There are different types, including:
     + **Character functions** (deal with text).
     + **Number functions** (deal with numbers).
     + **Date functions** (deal with dates).
     + **Conversion functions** (change one type of data to another).
2. **Multiple-Row Functions**
   * These functions work on groups of rows and give you one result for that whole group.
   * For example, they might calculate the total for all sales in a week.

**What Are Single-Row Functions?**

Single-row functions work on one row of data at a time and return one result for each row. Below are the different types of single-row functions: **Character Functions**, **Number Functions**, and **Date Functions**.

**1. Character Functions**

These functions deal with text (also called character data). They take text as input and can return either text or numbers.

**Types of Character Functions:**

* **Case Conversion Functions**: These change the case (uppercase or lowercase) of letters in text.
  + **LOWER(‘SQL Course’)** → sql course (all lowercase)
  + **UPPER(‘SQL Course’)** → SQL COURSE (all uppercase)
  + **INITCAP(‘SQL Course’)** → Sql Course (first letter capitalized)

**Example 1**:

* To print an employee's name with the first letter capitalized and their job title in lowercase:

sql

SELECT 'The job title for ' || INITCAP(ename) || ' is ' || LOWER(job) AS "EMPLOYEE DETAILS"

FROM emp;

**Example 2**:

* To display an employee's number, name (in uppercase), and department number for an employee named "Blake":

sql

SELECT empno, UPPER(ename), deptno

FROM emp

WHERE LOWER(ename) = 'blake';

Note: We convert the name to lowercase or uppercase when searching, just in case the name is stored differently.

**Character Manipulation Functions:**

These help in manipulating or working with text.

* **CONCAT(‘Good’, ‘String’)** → GoodString (joins two strings)
* **SUBSTR(‘String’, 2, 4)** → trin (extracts a part of the string starting from position 2, for 4 characters)
* **LENGTH(‘String’)** → 6 (gives the number of characters in the string)
* **INSTR(‘String’, ‘r’)** → 3 (finds the position of a letter in the string)
* **LPAD(sal, 10, ‘\*’)** → \*\*\*\*\*\*5000 (adds characters to the left of the text to make it 10 characters long)

**Example**:

* To show the employee name and job title combined, the length of the name, and the position of the letter "A" in their name for employees in sales:

sql

SELECT empno, CONCAT(ename, job), LENGTH(ename), INSTR(ename, 'A')

FROM emp

WHERE SUBSTR(job, 1, 5) = 'SALES';

**2. Number Functions**

Number functions work with numbers and return numeric results.

* **ROUND(column|expression, n)**: Rounds a number to a certain number of decimal places.
  + Example:
    - **ROUND(45.927, 2)** → 45.93 (rounds to 2 decimal places)
    - **ROUND(45.927)** → 46 (rounds to the nearest whole number)
    - **ROUND(45.927, -1)** → 50 (rounds to the nearest 10)
* **TRUNC(column|expression, n)**: Truncates a number (removes digits after the decimal point) to a certain number of decimal places.
  + Example:
    - **TRUNC(45.927, 2)** → 45.92 (cuts off after 2 decimal places)
    - **TRUNC(45.927)** → 45 (cuts off after the decimal point)
* **MOD(m, n)**: Returns the remainder when one number is divided by another.
  + Example:
    - **MOD(20, 3)** → 2 (20 divided by 3 leaves a remainder of 2)

**Examples**:

1. To round and truncate numbers:

sql

SELECT ROUND(45.923, 2), ROUND(45.923, 0), ROUND(45.923, -1)

FROM DUAL;

1. To calculate the remainder when dividing salary by commission for salesmen:

sql

SELECT ename, sal, comm, MOD(sal, comm)

FROM emp

WHERE UPPER(job) = 'SALESMAN';

**3. Date Functions**

* **SYSDATE**: Returns the current date and time.
  + To display the current date, we use the **DUAL** table, which is a dummy table in SQL that is used when we don’t need to use an actual table.

**Example**:

* To display the current date:

sql

SELECT SYSDATE

FROM DUAL;

That's a simplified explanation of single-row functions in SQL, focusing on character, number, and date functions. Let me know if anything is unclear!

**Arithmetic with Dates in SQL**

You can add or subtract numbers to dates in SQL to get a new date. For example, if you want to calculate how long employees in department 10 have been working (in weeks), you can subtract their hire date from the current date:

sql

SELECT ename, (SYSDATE - HIREDATE) / 7 AS "Number of Weeks"

FROM emp

WHERE deptno = 10;

**Date Functions**

Date functions help you work with dates in SQL. Most of these return a date value, except **MONTHS\_BETWEEN**, which returns a number.

**Common Date Functions:**

* **MONTHS\_BETWEEN(date1, date2)**: Returns the number of months between two dates.
  + Example: MONTHS\_BETWEEN('01-SEP-95', '11-JAN-94') → 19.68 months
* **ADD\_MONTHS(date, n)**: Adds n months to a date.
  + Example: ADD\_MONTHS('11-JAN-94', 6) → 11-JUL-94
* **NEXT\_DAY(date, 'DAY')**: Returns the next specific day after the given date.
  + Example: NEXT\_DAY('01-SEP-95', 'FRIDAY') → 08-SEP-95 (next Friday)
* **LAST\_DAY(date)**: Returns the last day of the month for a given date.
  + Example: LAST\_DAY('01-SEP-95') → 30-SEP-95
* **ROUND(date, 'format')**: Rounds the date to the nearest month or year.
  + Example: ROUND(TO\_DATE('25-JUL-95', 'DD-MON-YY'), 'MONTH') → 01-AUG-95
* **TRUNC(date, 'format')**: Truncates (cuts off) the date to the start of the month or year.
  + Example: TRUNC(TO\_DATE('25-JUL-95', 'DD-MON-YY'), 'MONTH') → 01-JUL-95

**Example 1:**

To display employees hired for less than 200 months, showing their hire date, tenure, review date, and more:

sql

SELECT empno, hiredate, MONTHS\_BETWEEN(SYSDATE, hiredate) AS TENURE, ADD\_MONTHS(hiredate, 6) AS REVIEW, NEXT\_DAY(hiredate, 'FRIDAY'), LAST\_DAY(hiredate)

FROM emp

WHERE MONTHS\_BETWEEN(SYSDATE, hiredate) < 2000;

**Example 2:**

To compare hire dates of employees who started in 1982:

sql

SELECT empno, hiredate, ROUND(hiredate, 'MONTH'), TRUNC(hiredate, 'MONTH')

FROM emp

WHERE hiredate LIKE '%82';

**Conversion Functions**

SQL has three main functions for converting data types:

1. **TO\_CHAR**: Converts a date or number into a string (text).
2. **TO\_NUMBER**: Converts a string into a number.
3. **TO\_DATE**: Converts a string into a date.

**Examples of TO\_CHAR with Dates:**

* **Example 1**: To display the employee number and the month and year of hiring:

sql

SELECT empno, TO\_CHAR(hiredate, 'MM/YY') AS Month\_Hired

FROM emp

WHERE ename = 'BLAKE';

* **Example 2**: To display employee names with the hire date formatted like "17 November, 1981":

sql

SELECT ename, TO\_CHAR(hiredate, 'fmDD Month YYYY') AS HIREDATE

FROM emp;

**Examples of TO\_CHAR with Numbers:**

* **Example 1**: To show the salary of an employee with a dollar sign:

sql

SELECT TO\_CHAR(sal, '$99,999') AS SALARY

FROM emp

WHERE ename = 'SCOTT';

**Example of TO\_DATE:**

* **Example**: To find employees who were hired on February 22, 1981:

sql

SELECT ename, hiredate

FROM emp

WHERE hiredate = TO\_DATE('February 22, 1981', 'Month dd, YYYY');

**DECODE Function**

The **DECODE** function works like an **IF-THEN-ELSE** statement. It checks an expression and returns a specific result based on conditions.

sql

DECODE(expression, search1, result1, search2, result2, ..., default)

**Example:**

To adjust salary based on the job title:

sql

SELECT job, sal, DECODE(job, 'ANALYST', sal \* 1.1, 'CLERK', sal \* 1.15, 'MANAGER', sal \* 1.20, sal) AS REVISED\_SALARY

FROM emp;

**Multiple-Row Functions (Group Functions)**

These functions work on groups of rows and return a single result per group.

**Examples of Group Functions:**

* **AVG**: Average value
* **MIN**: Minimum value
* **MAX**: Maximum value
* **COUNT**: Count of rows

**Example 1:**

To find the average, minimum, maximum salary, and number of employees:

sql

SELECT AVG(sal), MIN(sal), MAX(sal), COUNT(\*)

FROM emp;

**Example 2:**

To group employees by department and show the average salary per department:

sql

SELECT deptno, AVG(sal) AS AVERAGE\_SALARY

FROM emp

GROUP BY deptno;

**HAVING Clause**

The **HAVING** clause is used to filter groups, similar to how **WHERE** filters rows.

**Example:**

To show departments with an average salary above 2000, in descending order:

sql

SELECT deptno, AVG(sal), MAX(sal)

FROM emp

GROUP BY deptno

HAVING AVG(sal) > 2000

ORDER BY AVG(sal);

By using these functions, SQL allows you to perform complex calculations, format dates and numbers, and organize your data in flexible ways. Let me know if you need any further clarification!