LAB 4: Writing Single-row functions

Objectives

After completing this lesson, you should be able to do the following:

- ✓ Describe various types of functions available in SQL
- ✓ Use character, number, and date functions in SELECT statements
- ✓ Describe the use of conversion functions

Lesson Aim

Functions make the basic query block more powerful and are used to manipulate data values. This is the first of two lessons that explore functions. It focuses on singl e-row character, number, and date functions, as well as those functions that convert data from one type to another: for example, character data to numeric.

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

Features of single-row functions include:
• Acting on each row returned in the query
• Returning one result per row
• Possibly returning a data value of a different type than that referenced
• Possibly expecting one or more arguments
• Can be used in `SELECT`, `WHERE`, and `ORDER BY` clauses; can be nested
This lesson 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.)

Case Manipulation Functions:

Character Manipulation Functions

`CONCAT`, `SUBSTR`, `LENGTH`, `INSTR`, `LPAD`, `RPAD`, and `TRIM` are the character manipulation functions covered in this lesson.

- `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 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 quotes.)

Character-Manipulation Functions

These functions manipulate character strings:

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

1. SELECT empno, LOWER(ename), UPPER (ename),INITCAP (ename), LENGTH (ename), INSTR(ename, 'a') "Contains 'a '?" FROM emp;

Number Functions

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.

The DUAL Table

The DUAL table is owned by the user SYS and can be accessed by all users. It contains one column, DUMMY, and one row with the value X. The DUAL table is useful when you want to return a value once only: for instance, the value of a constant, pseudocolumn, or expression that is not derived from a table with user data. The DUAL table is generally used for SELECT clause syntax completeness, because both SELECT and FROM clauses are mandatory, and several calculations do not need to select from actual tables.

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.

MOD Function

The MOD function finds the remainder of value1 divided by value2. The slide example calculates the remainder of the salary after dividing it by 5,000 for all employees whose job ID is SA_REP.

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

2. SELECT ROUND(45.923,2), ROUND(45.923,0),
   ROUND(45.923,-1) FROM DUAL;
3. SELECT TRUNC(45.923,2), TRUNC(45.923),
   TRUNC(45.923,-2) FROM DUAL;
4.
```
SELECT last_name, salary, MOD(salary, 5000)
FROM employees
WHERE job_id = 'SA_REP';
```

Date Functions

• 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.

```
SELECT last_name, hire_date
FROM employees
WHERE last_name like 'G%';
```

The SYSDATE Function

SYSDATE is a date function that returns the current database server date and time. You can use SYSDATE just as you would use any other column name. For example, you can display the current date by selecting SYSDATE from a table. It is customary to select SYSDATE from a dummy table called DUAL.

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.

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.
• 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*.

5. SELECT SYSDATE FROM DUAL;
   SELECT ename, hiredate
   FROM emp;
6. SELECT ename, hiredate,

```
MONTHS_BETWEEN (SYSDATE, hiredate) ,
ADD_MONTHS (hiredate, 6) REVIEW,LAST_DAY(hiredate)
FROM emp;
```

NVL Function

Converts a null to an actual value

• Data types that can be used are date, character, and number.

• Data types must match:

- **NVL(commission_pct,0)**
- **NVL(hire_date,'01-JAN-97')**
- **NVL(job_id,'No Job Yet')**

The 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*.

Using the NVL Function

```
SELECT last_name, salary, NVL(commission_pct, 0),
(salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
      FROM employees;
```

Conditional Expressions

• Give you the use of IF-THEN-ELSE logic within a SQL statement. Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement:

```
SELECT last_name, job_id, salary,
      CASE job_id WHEN 'IT_PROG' THEN 1.10*salary
                  WHEN 'ST_CLERK' THEN 1.15*salary
                  WHEN 'SA_REP' THEN 1.20*salary
            ELSE salary END "REVISED_SALARY"
      FROM employees;
```

Using the CASE Expression

In the preceding SQL statement, the value of JOB_ID is decoded. If JOB_ID is IT_PROG, the salary increase is 10%; if JOB_ID is ST_CLERK, the salary increase is 15%; if JOB_ID is SA_REP, the salary increase is 20%. For all other job roles, there is no increase in salary.

Practice

1. Write a query that displays the employee's names with the first letter capitalized and all other letters lowercase and the length of the names.
2. Write a query to display the current date. Label the column Date

Date
08-MAR-01

3. For each employee, display the employee ID number, last_name, salary, and salary increased by 15% and expressed as a whole number. Label the column `New Salary`. Place your SQL statement in a text file named `lab3_2.sql`. Run your query in the file `lab3_2.sql`.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
100	King	24000	27600
101	Kochhar	17000	19550
102	De Haan	17000	19550
103	Hunold	9000	10350
104	Ernst	6000	6900

4. Modify your query `lab3_2.sql` to add a column that subtracts the old salary from the new salary. Label the column `Increase`. Save the contents of the file as `lab3_4.sql`. Run the revised query.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
100	King	24000	27600	3600
101	Kochhar	17000	19550	2550
102	De Haan	17000	19550	2550
103	Hunold	9000	10350	1350
104	Ernst	6000	6900	900
107	Lorentz	4200	4830	630
124	Mourgos	5800	6670	870
141	Rajs	3500	4025	525
142	Davies	3100	3565	465
143	Matos	2600	2990	390
144	Vargas	2500	2875	375
149	Zlotkey	10500	12075	1575

5. Write a query that displays the employee's last names with the first letter capitalized and all other letters lowercase and the length of the names, for all employees whose name starts with *J*, *A*,or *M*. Give each column an appropriate label. Sort the results by the employees' last names.

Name	Length
Abel	4
Matos	5
Mourgos	7

6. For each employee, display the employee's last name, and calculate the number of months between today and the date 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. (Note: Result may differ)

LAST_NAME	MONTHS_WORKED
Zlotkey	13
Mourgos	16
Grant	22
Lorentz	25
Vargas	32
Taylor	36
Matos	36
Fay	43
Davies	49
Abel	58
Hartstein	61
Rajs	65
Higgins	81
Gietz	81
LAST_NAME	MONTHS_WORKED
De Haan	98
Ernst	118
Hunold	134
Kochhar	138
Whalen	162
King	165

20 rows selected.

7. 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 similar to "Monday, the Thirty-First of July, 2000."

LAST_NAME	HIRE_DATE	REVIEW
King	17-JUN-87	Monday, the Twenty-First of December, 1987
Kochhar	21-SEP-89	Monday, the Twenty-Sixth of March, 1990
De Haan	13-JAN-93	Monday, the Nineteenth of July, 1993
Hunold	03-JAN-90	Monday, the Ninth of July, 1990
Ernst	21-MAY-91	Monday, the Twenty-Fifth of November, 1991
Lorentz	07-FEB-99	Monday, the Ninth of August, 1999
Mourgos	16-NOV-99	Monday, the Twenty-Second of May, 2000

8. Display the last name, hire date, and day of the week on which the employee started. Label the column `DAY`. Order the results by the day of the week starting with Monday.

LAST_NAME	HIRE_DATE	DAY
Grant	24-MAY-99	MONDAY
Ernst	21-MAY-91	TUESDAY
Mourgos	16-NOV-99	TUESDAY
Taylor	24-MAR-98	TUESDAY
Rajs	17-OCT-95	TUESDAY
Gietz	07-JUN-94	TUESDAY
Higgins	07-JUN-94	TUESDAY
King	17-JUN-87	WEDNESDAY
De Haan	13-JAN-93	WEDNESDAY
Davies	29-JAN-97	WEDNESDAY
Hunold	03-JAN-90	WEDNESDAY
Kochhar	21-SEP-89	THURSDAY
Whalen	17-SEP-87	THURSDAY
Vargas	09-JUL-98	THURSDAY

9. Create a query that displays the employees' last names and commission amounts. If an employee does not earn commission, put "No Commission." Label the column `COMM`.

LAST_NAME	COMM
King	No Commission
Kochhar	No Commission
De Haan	No Commission
Hunold	No Commission
Ernst	No Commission
Lorentz	No Commission
Mourgos	No Commission
Rajs	No Commission
Davies	No Commission
Matos	No Commission
Vargas	No Commission
Zlotkey	.2
Abel	.3
Taylor	.2