Lab2

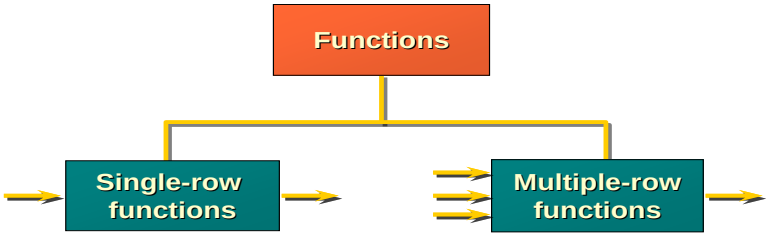
Objectives:

At the end of this lab, you should be able to

- Use some character functions.
- Use some number functions
- Converts date to char and vice versa.
- Use multiple row function :avg,sum,count,max,min

There are two distinct types of functions:

- Single-row functions
- Multiple-row functions



---

1. Single-row functions

a. **Character functions**: Accept character input and can return both character and number values
b. **Number functions**: Accept numeric input and return numeric values
c. **Date functions:** Operate on values of the date data type
d. **Conversion functions:** Convert a value from one data type to another
e. **General functions:** NVL function

---

a. Character functions:

i. **Lower(*column*|*expression*) :** converts alpha character values to lowercase.
ii. **Upper** (**(*column*|*expression*) :** converts alpha character values to uppercase.

Example : *select upper(ename),lower(ename) from emp;*

iii. **Length(*column*|*expression*):** returns number of characters in value.

Example : *select ename, length(ename) from emp;*

iv. **Substr(*column*|*expression*,f,n)**: returns characters from values starting at character position m, n characters long.

Example : *select ename, substr(ename,2,3) from emp;*

v. **Instr((*column*|*expression*,m) :** returns the numeric position of a named character.

Example : select ename, instr(ename,'A') from emp;

Prepared by : Saleh Alhazbi

Note: *The DUAL is a dummy table which 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.*

Example

SELECT INSTR('WELCOME','CO') FROM DUAL;

B) Number Functions

i. **ROUND(*column*|*expression*, *n*): r**ounds 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.)

Example : *SELECT ROUND(45.923,2), ROUND(45.923,0),ROUND(45.923,-1) FROM DUAL;*

ii. **TRUNC(column|expression,n):** Truncates the column, expression, or value to n decimal places or if n is omitted, no decimal places (If n is negative, numbers left of the decimal point are truncated to zero.)

Example : *SELECT TRUNC(45.923,2), TRUNC(45.923),TRUNC(45.923,-1) FROM DUAL;*

C) Date Functions

a. Oracle stores dates in an internal numeric format: century, year, month, day, hours, minutes, seconds.
b. The default date format is DD-MON-YY.
c. Arithmetic with Dates
   Date+number =Date (adds a number of days to a date)
   Date-number =Date (adds a number of days to a date)
   Date-date= number of days.

Note All date functions return a value of DATE data type except MONTHS_BETWEEN, which returns a numeric value.

i. **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.

Example : *Select ename,months_between(sysdate,hiredate) from emp;*

ii. **ADD_MONTHS(date, n):** Adds *n* number of calendar months to *date*. The value of *n* must be an integer and can be negative.

Example : *Select ename, ADD_MONTHS(hiredate, 6) REVIEW from emp;*

Practice:

1) For each employee display the employee 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.

D) Data type Conversion

SQL provides three functions to convert a value from one datatype to another:

i. **TO_CHAR(*date, 'fmt'*)**

Example : *SELECT ename, TO_CHAR(hiredate, 'DD Month YYYY') FROM emp;*

Display the name, hire date, and day of the week on which the employee started. Label the column DAY.
SELECT ENAME, HIREDATE, TO_CHAR(HIREDATE, 'DAY') FROM EMP;

Oracle TO_CHAR supports the following format specifiers for datetime values:

Oracle TO_CHAR	Format Specifier
YYYY	4-digit year
YY	2-digit year
MON	Abbreviated month (Jan - Dec)
MONTH	Month name (January - December)
MM	Month (1 - 12)
DY	Abbreviated day (Sun - Sat)
DD	Day (1 - 31)
D	Day of the week in number(1..7)
HH24	Hour (0 - 23)
HH or HH12	Hour (1 - 12)
MI	Minutes (0 - 59)
SS	Seconds (0 - 59)

ii. **TO_NUMBER(*char*[, '*fmt*'])**
Example : SELECT TO_NUMBER('123')+4 FROM DUAL;

iii. **TO_DATE(*char, 'fmt'*)**
Example : SELECT ename, hiredate FROM emp WHERE hiredate = TO_DATE('February 22, 1981', 'Month DD, YYYY');

E) NVL Function

Converts null to an actual value

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

Example: SELECT ename, sal, comm, (sal*12)+NVL(comm,0) FROM emp;

Practice:

Create a query that will display the employee name and commission amount. If the employee does not earn commission, put "No Commission." Label the column COMM.

2- Multiple-row functions:

Note:

- You can use MIN and MAX for any datatype.
- Group functions ignore null values in the column.

example:

```
SQL> SELECT    AVG(sal), MAX(sal),MIN(sal), SUM(sal)    FROM emp
2    WHERE     job LIKE 'SALES%';

SQL> SELECT    COUNT(*)  FROM    emp  WHERE deptno = 30;

SQL> SELECT    COUNT(comm)  FROM       emp  WHERE deptno = 30;

SQL> SELECT AVG(comm)  FROM   emp;

SQL> SELECT AVG(NVL(comm,0))  FROM   emp;
```

Note the difference

Note the difference

Creating Groups of Data

Divide rows in a table into smaller groups by using the GROUP BY clause

```
SELECT      column, group_function(column)
FROM              table
[WHERE      condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

Example:

```
SQL> SELECT   deptno, AVG(sal)  FROM     emp  GROUP BY deptno;
```

The HAVING Clause

- **You cannot use the WHERE clause to restrict groups.**
- The Oracle Server performs the following steps when you use the HAVING clause:
  - Rows are grouped.
  - The group function is applied to the group.

- The groups that match the criteria in the HAVING clause are displayed.

Example:

```
SQL>SELECT    job, SUM(sal) from emp group by job
 2      having sum(sal)<5000 ;

SQL> SELECT    job, SUM(sal) PAYROLL FROM  emp  WHERE  job NOT LIKE 'SALES%'
2    GROUP BY  job  ORDER BY  SUM(sal);
```

Practice

1. Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number.

2. Display the minimum, maximum, sum, and average salary for each job type.

3. Write a query to display the number of people with the same job.