Functions in SQL

Number Functions

Number functions accept numeric input and return numeric values. Most of these functions return values that are accurate to 38 decimal digits.

Some of the number functions available in Oracle are:

ABS

ABS returns the absolute value of *n*.

The following example returns the absolute value of -87:

```
SELECT ABS(-87) "Absolute" FROM DUAL;
```

```
 Absolute
----------
        87
```

ACOS

ACOS returns the arc cosine of *n*. Inputs are in the range of -1 to 1, and outputs are in the range of 0 to pi and are expressed in radians.

The following example returns the arc cosine of .3:

```
SELECT ACOS(.3)"Arc_Cosine" FROM DUAL;
```

```
Arc_Cosine
----------
1.26610367
```

Similar to ACOS, you have ASIN (Arc Sine), ATAN (Arc Tangent) functions.

CIEL

Returns the lowest integer above the given number.

Example:
The following function return the lowest integer above 3.456;

```
select ciel(3.456) “Ciel” from dual;
```

```
Ciel
```

```
---------
        4
```

FLOOR

Returns the highest integer below the given number.

Example:

The following function return the highest integer below 3.456;

select floor(3.456) “Floor” from dual;

```
Floor
------------
        3
```

COS

Returns the cosine of an angle (in radians).

Example:

The following example returns the COSINE angle of 60 radians.

```
select cos(60) “Cosine” from dual;
```

SIN

Returns the Sine of an angle (in radians).

Example:

The following example returns the SINE angle of 60 radians.

```
select  SIN(60) “Sine” from dual;
```

TAN

Returns the Tangent of an angle (in radians).

Example:

The following example returns the tangent angle of 60 radians.

```
select Tan(60) “Tangent” from dual;
```

Similar to SIN, COS, TAN functions hyperbolic functions SINH, COSH, TANH are also available in oracle.

MOD

Returns the remainder after dividing m with n.
Example

The following example returns the remainder after dividing 30 by 4.

```
Select mod(30,4) “MOD” from dual;

MOD
---------
        2
```

POWER

Returns the power of m, raised to n.
Example
The following example returns the 2 raised to the power of 3.

```
select  power(2,3) “Power” from dual;

POWER
---------
        8
```

LN

Returns natural logarithm of n.
Example
The following example returns the natural logarithm of 2.

```
select ln(2) from dual;
LN
------------
```

LOG

Returns the logarithm, base m, of n.
Example
The following example returns the log of 100.

```
select log(10,100) from dual;
LOG
---------
        2
```

ROUND

Returns a decimal number rounded of to a given decimal positions.

Example

The following example returns the no. 3.4573 rounded to 2 decimals.

```
 select round(3.4573,2) “Round” from dual;
```

```
Round
------------
        3.46
```

TRUNC

Returns a decimal number Truncated to a given decimal positions.

Example

The following example returns the no. 3.4573 truncated to 2 decimals.

```
select round(3.4573,2) “Round” from dual;
```

```
Round
------------
        3.45
```

SQRT

Returns the square root of a given number.

Example

The following example returns the square root of 16.

```
select sqrt(16) from dual;
```

```
SQRT
---------
        4
```

Aggregate Functions

Aggregate functions return a single value based on groups of rows, rather than single value for each row. You can use Aggregate functions in select lists and in ORDER BY and HAVING clauses. They are commonly used with the GROUP BY clause in a SELECT statement, where Oracle divides the rows of a queried table or view into groups.

The important Aggregate functions are :

Avg Sum Max Min Count Stddev Variance

AVG

AVG(ALL /DISTINCT expr)

Returns the average value of expr.

Example

The following query returns the average salary of all employees.

```
select avg(sal) “Average Salary” from emp;
```

```
Average Salary
------------------------
2400.40
```

SUM

SUM(ALL/DISTINCT expr)

Returns the sum value of expr.

Example

The following query returns the sum salary of all employees.

```
select sum(sal) “Total Salary” from emp;
```

```
Total Salary
------------------------
26500
```

MAX

MAX(ALL/DISTINCT expr)

Returns maximum value of expr.

Example

The following query returns the max salary from the employees.

```
select max(sal) “Max Salary” from emp;
```

```
Maximum Salary
------------------------
4500
```

MIN

MIN(ALL/DISTINCT expr)

Returns minimum value of expr.

Example

The following query returns the minimum salary from the employees.

```
select min(sal) “Min Salary” from emp;
```

```
Minimum Salary
------------------------
1200
```

COUNT

COUNT(*) OR COUNT(ALL/DISTINCT expr)

Returns the number of rows in the query. If you specify expr then count ignore nulls. If you specify the asterisk (*), this function returns all rows, including duplicates and nulls. COUNT never returns null.

Example

The following query returns the number of employees.

```
Select count(*) from emp;
```

```
COUNT
------
14
```

The following query counts the number of employees whose salary is not null.

```
Select count(sal) from emp;
```

```
COUNT
------
12
```

STDDEV

STDDEV(ALL/DISTINCT   expr)

STDDEV returns sample standard deviation of *expr*, a set of numbers.

Example

The following query returns the standard deviation of salaries.

```
select stddev(sal) from emp;
```

```
Stddev
-------
1430
```

VARIANCE

VARIANCE(ALL/DISTINCT   expr)

Variance returns the variance of *expr*.

Example

The following query returns the variance of salaries.

```
select variance(sal) from emp;
```

```
Variance
-------
1430
```

Character Functions

Character functions operate on values of dataype CHAR or VARCHAR.

LOWER

Returns a given string in lower case.

```
select LOWER('SAMI') from dual;
```

```
LOWER
-------------
sami
```

UPPER

Returns a given string in UPPER case.

```
select UPPER('Sami') from dual;
```

```
UPPER
------------------
SAMI
```

INITCAP

Returns a given string with Initial letter in capital.

```
select INITCAP('mohammed sami') from dual;
```

```
INITCAP
------------------
Mohammed Sami
```

LENGTH

Returns the length of a given string.

```
select length('mohammed sami') from dual;
```

```
LENGTH
------------
  13
```

SUBSTR

Returns a substring from a given string. Starting from position p to n characters.
For example the following query returns “sam” from the string “mohammed sami”.

```
select substr('mohammed sami',10,3) from dual;
```

```
Substr
--------
sam
```

INSTR

Tests whether a given character occurs in the given string or not. If the character occurs in the string then returns the first position of its occurrence otherwise returns 0.

Example

The following query tests whether the character “a” occurs in string “mohammed sami”

```
select instr('mohammed sami','a') from dual;
```

```
INSTR
--------
4
```

REPLACE

Replaces a given set of characters in a string with another set of characters.

Example

The following query replaces “mohd” with “mohammed” .

```
select replace('ali mohd khan','mohd','mohammed') from
dual;
```

```
REPLACE
---------
ali mohammed khan
```

REPLACE

Replaces a given set of characters in a string with another set of characters.

Example

The following query replaces “mohd” with “mohammed” .

```
select replace('ali mohd khan','mohd','mohammed') from
dual;
```

```
REPLACE
---------
ali mohammed khan
```

TRANSLATE

This function is used to encrypt characters. For example you can use this function to replace characters in a given string with your coded characters.

Example

The following query replaces characters A with B, B with C, C with D, D with E,...Z with A, and a with b,b with c,c with d, d with e ....z with a.

```
select
translate('interface','ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghi
jklmnopqrstuvwxyz',

'BCDEFGHIJKLMNOPQRSTUVWXYZAbcdefghijklmnopqrstuvwxyza')
"Encrypt" from dual;



Encrypt
-----------
joufsgbdf
```

SOUNDEX

This function is used to check pronounciation rather than exact characters. For example many people write names as “smith” or “smyth” or “smythe” but they are pronounced as smith only.

Example

The following example compare those names which are spelled differently but are pronouced as “smith”.

```
Select ename from emp where
soundex(ename)=soundex('smith');

ENAME
---------
Smith
Smyth
Smythe
```

RPAD

Right pads a given string with a given character to n number of characters.

Example

The following query rights pad ename with '*' until it becomes 10 characters.

```
select rpad(ename,'*',10) from emp;
```

```
Ename
----------
Smith*****
John******
Mohammed**
Sami******
```

LPAD

Left pads a given string with a given character upto n number of characters.

Example

The following query left pads ename with '*' until it becomes 10 characters.

```
select lpad(ename,'*',10) from emp;
Ename
----------
*****Smith
******John
**Mohammed
******Sami
```

LTRIM

Trims blank spaces from a given string from left.

Example

The following query returns string “   Interface   “ left trimmed.

```
select ltrim('       Interface       ') from dual;
Ltrim
--------------
Interface
```

RTRIM

Trims blank spaces from a given string from Right.

Example

The following query returns string “   Interface   “ right trimmed.

```
select rtrim('       Interface       ') from dual;

Rtrim
------------
   Interface
```

TRIM

Trims a given character from left or right or both from a given string.

Example

The following query removes zero from left and right of a given string.

```
Select trim('0′ from '00003443500') from dual;
```

```
Trim
----------
34435
```

CONCAT

Combines a given string with another string.

Example

The following Query combines ename with literal string " is a " and jobid.

```
Select concat(concat(ename,' is a '),job) from emp;
```

```
Concat
----------------
Smith is a clerk
John is a Manager
Sami is a G.Manager
```

Miscellaneous Single Row Functions

DECODE

DECODE(expr, searchvalue1, result1,searchvalue2,result2,..., defaultvalue)

Decode functions compares an expr with search value one by one. If the expr does not match any of the search value then returns the default value. If the default value is omitted then returns null.

Example

The following query returns the department names according the deptno. If the deptno does not match any of the search value then returns "Unknown Department"

```
select
decode(deptno,10,'Sales',20,'Accounts,30,'Production′,

40,'R&D','Unknown Dept') As DeptName from emp;
```

```
DEPTNAME
```

```
----------
Sales
Accounts
Unknown Dept.
Accounts
Production
Sales
R&D
Unknown Dept.
```

GREATEST

GREATEST(expr1, expr2, expr3,expr4...)

Returns the greatest expr from a expr list.

Example

```
select greatest(10,20,50,20,30) from dual;

GREATEST
--------
50

select greatest('SAMI','SCOTT','RAVI','SMITH','TANYA')
from dual;

GREATEST
--------
TANYA
```

LEAST

LEAST(expr1, expr2, expr3,expr4...)

It is simillar to greatest. It returns the least expr from the expression list.

```
select least(10,20,50,20,30) from dual;

LEAST
--------
10

select least('SAMI','SCOTT','RAVI','SMITH','TANYA') from
dual;
```

```
LEAST
--------
RAVI
```

NVL

NVL(expr1,expr2)

This function is oftenly used to check null values. It returns expr2 if the expr1 is null, otherwise returns expr1.

Example

The following query returns commission if commission is null then returns 'Not Applicable'.

```
Select ename,nvl(comm,'Not Applicable') “Comm” from dual;
```

```
ENAME      COMM
------     ----
Scott      300
Tiger      450
Sami       Not Applicable
Ravi       300
Tanya      Not Applicable
```

Date Functions and Operators.

To see the system date and time use the following functions :

CURRENT_DATE :returns the current date in the session time zone, in a value in the Gregorian calendar of datatype `DATE`

SYSDATE :Returns the current date and time.

SYSTIMESTAMP :The `SYSTIMESTAMP` function returns the system date, including fractional seconds and time zone of the database. The return type is `TIMESTAMP WITH TIME ZONE`.

SYSDATE Example

To see the current system date and time give the following query.

```
select sysdate from dual;

SYSDATE
-------
21-JUL-15
```

The format in which the date is displayed depends on NLS_DATE_FORMAT parameter.

For example set the NLS_DATE_FORMAT to the following format

```
Alter session set NLS_DATE_FORMAT=’DD-MON-YYYY HH:MIpm’;
```

Then give the give the following statement

```
select sysdate from dual;

SYSDATE
------------------
21-JUL-2003 03:15pm
```

The default setting of NLS_DATE_FORMAT is DD-MON-YY

CURRENT_DATE Example

To see the current system date and time with time zone use CURRENT_DATE function

```
ALTER SESSION SET TIME_ZONE = '-4:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY
HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;

SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-04:00          22-APR-2003 14:15:03

ALTER SESSION SET TIME_ZONE = '-7:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;

SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-07:00          22-APR-2003 09:15:33
```

SYSTIMESTAMP Example

To see the current system date and time with fractional seconds with time zone give the following statement

```
Select systimestamp from dual;

SYSTIMESTAMP
-------------------------------
22-APR-03 08.38.55.538741 AM -07:00
```

DATE FORMAT MODELS

To translate the date into a different format string you can use TO_CHAR function with date format. For example to see the current day you can give the following query

To translate a character value, which is in format other than the default date format, into a date value you can use TO_DATE function with date format to date

```
 Select to_char(sysdate,'DAY')"Today" FROM DUAL;

TODAY
-------
THURSDAY
```

Like this "DAY" format model there are many other date format models available in Oracle. The following table list date format models.

FORMAT	**MEANING**
D	Day of the week
DD	Day of the month
DDD	Day of the year
DAY	Full day for ex. 'Monday', 'Tuesday', 'Wednesday'
DY	Day in three letters for ex. 'MON', 'TUE','FRI'
W	Week of the month
WW	Week of the year
MM	Month in two digits (1-Jan, 2-Feb,…12-Dec)
MON	Month in three characters like "Jan", "Feb", "Apr"
MONTH	Full Month like "January", "February", "April"
RM	Month in Roman Characters (I-XII, I-Jan, II-Feb,…XII-Dec)
Q	Quarter of the Month

YY	Last two digits of the year.
YYYY	Full year
YEAR	Year in words like “Nineteen Ninety Nine”
HH	Hours in 12 hour format
HH12	Hours in 12 hour format
HH24	Hours in 24 hour format
MI	Minutes
SS	Seconds
FF	Fractional Seconds
SSSSS	Milliseconds
J	Julian Day i.e Days since 1st-Jan-4712BC to till-date
RR	If the year is less than 50 Assumes the year as 21ST Century. If the year is greater than 50 then assumes the year in 20th Century.

FORMAT	**MEANING**
D	Day of the week
DD	Day of the month
DDD	Day of the year
DAY	Full day for ex. ‘Monday’, ’Tuesday’, ’Wednesday’
DY	Day in three letters for ex. ‘MON’, ‘TUE’,’FRI’
W	Week of the month
WW	Week of the year
MM	Month in two digits (1-Jan, 2-Feb,…12-Dec)
MON	Month in three characters like “Jan”, ”Feb”, ”Apr”
MONTH	Full Month like “January”, ”February”, ”April”
RM	Month in Roman Characters (I-XII, I-Jan, II-Feb,…XII-Dec)
Q	Quarter of the Month
YY	Last two digits of the year.
YYYY	Full year
YEAR	Year in words like “Nineteen Ninety Nine”
HH	Hours in 12 hour format
HH12	Hours in 12 hour format
HH24	Hours in 24 hour format
MI	Minutes
SS	Seconds
FF	Fractional Seconds
SSSSS	Milliseconds
J	Julian Day i.e Days since 1st-Jan-4712BC to till-date
RR	If the year is less than 50 Assumes the year as 21ST Century. If the year is greater than 50 then assumes the year in 20th Century.

SUFFIXES

TH	Returns th, st, rd or nd according to the leading number like 1st , 2nd 3rd 4th
SP	Spells out the leading number
AM or PM	Returns AM or PM according to the time
SPTH	Returns Spelled Ordinal number. For. Example First, Fourth

For example to see the today's date in the following format

Friday, 7th March, 2014

Give the following statement

```
select to_char(sysdate,'Day, ddth Month, yyyy')"Today"
from dual;

TODAY
------------------------
Friday, 7th March, 2014
```

For example you want to see hire dates of all employee in the following format

Friday, 8th August, 2003

Then give the following query.

```
select to_char(hire_date,'Day, ddth Month, yyyy') from
emp;
```

TO_DATE Example

To_Date function is used to convert strings into date values. For example you want to see what was the day on 15-aug-1947. The use the to_date function to first convert the string into date value and then pass on this value to to_char function to extract day.

```
select to_char(to_date('15-aug-1947','dd-mon-yyyy'),'Day')
                                          from dual;

TO_CHAR(
--------
Friday
```

To see how many days have passed since 15-aug-1947 then give the following query

```
Select sysdate-to_date('15-aug-1947','dd-mon-yyyy')
                              from dual;
```

Now we want to see which date will occur after 45 days from now

```
Select sysdate+45 from dual;

SYSDATE
-------
06-JUN-2003
```

ADD_MONTHS

To see which date will occur after 6 months from now, we can use ADD_MONTHS function

```
Select ADD_MONTHS(SYSDATE,6) from dual;

ADD_MONTHS
----------
22-OCT-2003
```

MONTHS_BETWEEN

To see how many months have passed since 15-aug-1947, use the MONTHS_BETWEEN function.

```
Select months_between(sysdate,to_date('15-aug-1947'))
                                  from dual;

Months
------
616.553
```

To eliminate the decimal value use truncate function

LAST_DAY

To see the last date of the month of a given date, Use LAST_DAY function.

```
select LAST_DAY(sysdate) from dual;

LAST_DAY
--------
31-AUG-2003
```

NEXT_DAY

To see when the next Saturday is coming, use the NEXT_DAY function.

```
select next_day(sysdate) from dual;

NEXT_DAY
-----------
09-AUG-2003
```

EXTRACT

An EXTRACT datetime function extracts and returns the value of a specified datetime field from a datetime or interval value expression. When you extract a TIMEZONE_REGION or TIMEZONE_ABBR (abbreviation), the value returned is a string containing the appropriate time zone name or abbreviation

The syntax of EXTRACT function is

EXTRACT (YEAR / MONTH / WEEK / DAY / HOUR / MINUTE / TIMEZONE FROM DATE)
Example

The following demonstrate the usage of EXTRACT function to extract year from current date.

```
Select extract(year from sysdate) from dual;
 EXTRACT
-------
2015
```

NEW_TIME( )

The following example returns an Atlantic Standard time, given the Pacific Standard time equivalent:

```
ALTER SESSION SET NLS_DATE_FORMAT =  'DD-MON-YYYY
HH24:MI:SS';
```

```
SELECT NEW_TIME(TO_DATE('11-10-99 01:23:45', 'MM-DD-YY
HH24:MI:SS'),
   'AST', 'PST') "New Date and Time" FROM DUAL;

New Date and Time
--------------------
09-NOV-1999 21:23:45
```

ALL, ANY Comparison Conditions in SQL

```
SQL> SELECT * FROM emp;

EMPNO ENAME      JOB         MGR HIREDATE          SAL
COMM   DEPTNO
---------- ---------- --------- ---------- ---------------
----- ---------- ---------- ----------
7369 SMITH      CLERK       7902 17-DEC-1980 00:00:00
800              20
7499 ALLEN      SALESMAN     7698 20-FEB-1981 00:00:00
1600  300        30
7521 WARD       SALESMAN     7698 22-FEB-1981 00:00:00
1250  500        30
7566 JONES      MANAGER      7839 02-APR-1981 00:00:00
2975             20
7654 MARTIN     SALESMAN         7698 28-SEP-1981 00:00:00
1250  1400       30
7698 BLAKE      MANAGER          7839 01-MAY-1981 00:00:00
2850             30
7782 CLARK      MANAGER          7839 09-JUN-1981 00:00:00
2450             10
7788 SCOTT      ANALYST          7566 19-APR-1987 00:00:00
3000             20
7839 KING       PRESIDENT     17-NOV-1981 00:00:00
5000             10
      7844 TURNER     SALESMAN         7698 08-SEP-1981
00:00:00       1500       0       30
      7876 ADAMS      CLERK      7788 23-MAY-1987 00:00:00
1100             20
      7900 JAMES      CLERK      7698 03-DEC-1981 00:00:00
950              30
      7902 FORD       ANALYST          7566 03-DEC-1981
00:00:00       3000              20
```

```
      7934 MILLER     CLERK       7782 23-JAN-1982 00:00:00
1300             10

SQL>
```

- ALL
- ANY

ALL

The ALL comparison condition is used to compare a value to a list or subquery. It must be preceded by =, !=, >, <, <=, >= and followed by a list or subquery.

When the ALL condition is followed by a list, the optimizer expands the initial condition to all elements of the list and strings them together with AND operators, as shown below.

```
SELECT empno, sal
FROM   emp
WHERE  sal > ALL (2000, 3000, 4000);

     EMPNO        SAL
---------- ----------
      7839       5000

SQL>

-- Transformed to equivalent statement without ALL.

SELECT empno, sal
FROM   emp
WHERE  sal > 2000 AND sal > 3000 AND sal > 4000;

     EMPNO        SAL
---------- ----------
      7839       5000

SQL>
```

When the ALL condition is followed by a subquery, the optimizer performs a two-step transformation as shown below.

```
SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  e1.sal > ALL (SELECT e2.sal
                     FROM   emp e2
                     WHERE  e2.deptno = 20);

     EMPNO        SAL
---------- ----------
      7839       5000

SQL>

-- Transformed to equivalent statement using ANY.

SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  NOT (e1.sal <= ANY (SELECT e2.sal
                           FROM emp e2
                           WHERE e2.deptno = 20));

     EMPNO        SAL
---------- ----------
      7839       5000

SQL>

-- Transformed to equivalent statement without ANY.

SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  NOT EXISTS (SELECT e2.sal
                   FROM emp e2
                   WHERE e2.deptno = 20
                   AND   e1.sal <= e2.sal);

     EMPNO        SAL
---------- ----------
      7839       5000

SQL>
```

Assuming subqueries don't return zero rows, the following statements can be made for both list and subquery versions:/p>

- "x = ALL (...)": The value must match all the values in the list to evaluate to TRUE.
- "x != ALL (...)": The value must not match any values in the list to evaluate to TRUE.
- "x > ALL (...)": The value must be greater than the biggest value in the list to evaluate to TRUE.
- "x < ALL (...)": The value must be smaller than the smallest value in the list to evaluate to TRUE.
- "x >= ALL (...)": The value must be greater than or equal to the biggest value in the list to evaluate to TRUE.
- "x <= ALL (...)": The value must be smaller than or equal to the smallest value in the list to evaluate to TRUE.

If a subquery returns zero rows, the condition evaluates to TRUE.

```
SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  e1.sal > ALL (SELECT e2.sal FROM emp e2 WHERE
e2.deptno = 100);

     EMPNO        SAL
---------- ----------
      7369        800
      7900        950
      7876       1100
      7521       1250
      7654       1250
      7934       1300
      7844       1500
      7499       1600
      7782       2450
      7698       2850
      7566       2975
      7788       3000
      7902       3000
      7839       5000

SQL>
```

ANY

The ANY comparison condition is used to compare a value to a list or subquery. It must be preceded by =, !=, >, <, <=, >= and followed by a list or subquery.

When the ANY condition is followed by a list, the optimizer expands the initial condition to all elements of the list and strings them together with OR operators, as shown below.

```
SELECT empno, sal
FROM   emp
WHERE  sal > ANY (2000, 3000, 4000);

     EMPNO        SAL
---------- ----------
      7566       2975
      7698       2850
      7782       2450
      7788       3000
      7839       5000
      7902       3000

SQL>

-- Transformed to equivalent statement without ANY.

SELECT empno, sal
FROM   emp
WHERE  sal > 2000 OR sal > 3000 OR sal > 4000;

     EMPNO        SAL
---------- ----------
      7566       2975
      7698       2850
      7782       2450
      7788       3000
      7839       5000
      7902       3000

SQL>
```

When the ANY condition is followed by a subquery, the optimizer performs a single transformation as shown below.

```
SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  e1.sal > ANY (SELECT e2.sal
                     FROM   emp e2
                     WHERE  e2.deptno = 10);

     EMPNO        SAL
---------- ----------
      7839       5000
      7902       3000
      7788       3000
      7566       2975
      7698       2850
      7782       2450
      7499       1600
      7844       1500

SQL>

-- Transformed to equivalent statement without ANY.

SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  EXISTS (SELECT e2.sal
               FROM emp e2
               WHERE e2.deptno = 10
               AND   e1.sal > e2.sal);

     EMPNO        SAL
---------- ----------
      7839       5000
      7902       3000
      7788       3000
      7566       2975
      7698       2850
      7782       2450
      7499       1600
      7844       1500

SQL>
```

Assuming subqueries don't return zero rows, the following statements can be made for both list and subquery versions:

- "x = ANY (...)": The value must match one or more values in the list to evaluate to TRUE.
- "x != ANY (...)": The value must not match one or more values in the list to evaluate to TRUE.
- "x > ANY (...)": The value must be greater than the smallest value in the list to evaluate to TRUE.
- "x < ANY (...)": The value must be smaller than the biggest value in the list to evaluate to TRUE.
- "x >= ANY (...)": The value must be greater than or equal to the smallest value in the list to evaluate to TRUE.
- "x <= ANY (...)": The value must be smaller than or equal to the biggest value in the list to evaluate to TRUE.

If a subquery returns zero rows, the condition evaluates to FALSE.

```
SELECT e1.empno, e1.sal
FROM   emp e1
WHERE  e1.sal > ANY (SELECT e2.sal FROM emp e2 WHERE
e2.deptno = 100);

no rows selected
```