6. ARITHMETIC AND DATE FUNCTIONS

- What is a function?
- Types of functions
- Arithmetic functions
- Dual table
- Date arithmetic
- Date functions
- Summary
- Exercises

What is a function?

A function is similar to an operator in operation. A function is a name that performs a specific task. A function may or may not take values (arguments) but it always returns a value as the result. If function takes values then these values are to be given within parentheses after the function name. The following is the general format of a function.

```
function [(argument-1, argument-2,...) ]
```

If the function doesn't take any value then function name can be used alone and even parentheses are not required.

Types of functions

Functions are classified based on the type of data on which they perform the operation. The following are the different types of functions available in Oracle.

- Arithmetic Functions.
- Date & Time functions.
- String functions.
- Conversion functions.
- Miscellaneous functions.
- Group functions.

Arithmetic functions perform take numeric data; date functions take date type data and string functions take strings. Conversion functions are used to convert the given value from one type to another. Miscellaneous functions perform operations on any type of data. Group functions are used to perform operations on the groups created by GROUP BY clause.

***Note**: Group functions or aggregate functions perform their operation on a group (a collection of rows). All the remaining functions are called as* single-row *functions as they return a result for each row.*

Arithmetic Functions

Arithmetic functions take numbers and perform arithmetic operations. Table 1 lists the arithmetic functions available in Oracle.

Function	Description
ABS(value)	Absolute value of the given value.
CEIL(value)	Smallest integer larger than or equal to value
FLOOR(value)	Largest integer smaller than or equal to value
MOD(value,divisor)	Remainder of the division between the value and divisor.
POWER(value,exponent)	Value is raised to exponent.
ROUND(value[,precision])	Rounds value to precision. Precision can be negative if rounding is to be done on the left of the decimal point.
TRUNC(value[,precision])	Truncates instead of rounding. Otherwise same as ROUND.
SQRT(value)	Square root of value.
SIGN(value)	Returns 1 if value > 0, -1 if value <0, 0 if value = 0

Table 1: Arithmetic Functions.

The following are a few examples of arithmetic functions.

```
select mod(10,4) from dual;
```

The above command displays 2 as the result as the remainder of the division between 10 and 4 is 2.

ROUND and TRUNC functions

ROUND and TRUNC functions are used to round and truncate the given number to the given number of digits (either on the right or left of the decimal point). ROUND takes the leftmost digit that is being lost and accordingly adds one to the rightmost digit. TRUNC doesn't take the leftmost digit into account. It just truncates the given number to the given precision.

```
select round(1047.785,2), trunc(1047.785,2) from dual;
```

The above command will display 1047.79 and 1047.78. This is because TRUNC doesn't take the digits being lost into account.

The following examples illustrate the result of ROUND and TRUNC functions with positive and negative precision.

Function	Result
ROUND(1295.356,2)	1295.36
TRUNC(1295.356,2)	1295.35
ROUND(1295.356,0)	1295
ROUND(1285.356, -1)	1290
TRUNC(1285.356, -1)	1280
ROUND(1295,-2)	1300

When precision in ROUND is positive then rounding takes place to the specified number of digits on the right of decimal point. For example, if precision is 2 it means round number to 2 digits on the right of the decimal point. When precision is negative; it means number is to be rounded to the left of the decimal point. In both the cases if the leftmost digit of the digits being lost is >= 5 then one is added to the rightmost digit of the digits that are retained.

In the example given above, ROUND (1285.356, -1) will result in 1290. Because one digit on the left of decimal point is to be set to zero, means 5 is replaced with 0. As the digit that is replaced with zero (5 in this case) is >= 5 one is added to digit on the left of it (8) to make it 9. So the result is 1290.

ROUND (1295,-2) results in 1300. This is because in 1295 two digits on the left of decimal point are set to zeroes(1200) and as leftmost digit out of digits that are set to zero is 9 one is added to 2, which is the rightmost digit in the remaining portion. This makes it 1300.

***Note**: ROUND and TRUNC functions can also be used with **date** data type. More on this later in this chapter.*

CEIL & FLOOR functions

CEIL produces the smallest integer that is greater than or equal to the given value. Whereas FLOOR is opposite of CEIL. The following table illustrates the usage of these two related functions.

Function	Result
CEIL(1.3)	2
CEIL(2)	2
CEIL(-2.3)	-2
FLOOR(1.3)	1
FLOOR(2)	2
FLOOR(-2.3)	-3

DUAL Table

This is a table that is made available to every account in Oracle database. This table contains one row and one column. This table can be used with SELECT when result of the expression is to be displayed only for once.

```
SQL> describe dual
 Name                      Null?    Type
 ------------------------- -------- ----
 DUMMY                              VARCHAR2(1)
```

For example, to display the current system date the following SELECT can be used:

```
SQL> select sysdate from dual;

SYSDATE
--------
24-AUG-01
```

As DUAL table contains only one row, the result is displayed only for once.

The following example displays the course fee by rounding it to thousands.

```
select ccode,name, round(fee,-3) "fee"
from courses;

CCODE NAME                              fee
----- ------------------------------ ---------
ora   Oracle database                 5000
vbnet VB.NET                          6000
c     C programming                   4000
asp   ASP.NET                         5000
java  Java Language                   5000
xml   XML Programming                 4000
```

Scientific functions

The following are the other arithmetic functions that are rarely used in business applications. They are mainly used in scientific applications. However as they are available in Oracle, we will get to know them.

Function	Description
ACOS(value)	Arc cosine of the given *value*. The value is in the range –1 to 1 and return value is in radians.
ASIN(value)	Arc sine of the *value*.
ATAN(value)	Arc tangent of the *value*.
COS(value)	Cosine of the *value*.
COSH(value)	Hyperbolic cosine of the *value*.
EXP(value)	Return **e** (2.71828183) raised to *value* power.
LN(value)	Natural logarithm of *value*.
LOG(base,value)	Logarithm with *base* of *value*.
SIN(value)	Sine of the *value*.
SINH(value)	Hyperbolic sine of the *value*.
TAN(value)	Tangent of the *value*.
TANH(value)	Hyperbolic tangent of the *value*.

Table 2: Scientific Functions.

The following are few examples of these scientific functions.

```
select exp(2) from dual;

   EXP(2)
---------
7.3890561


select log(10,10) from dual

LOG(10,10)
----------
         1
```

```
SELECT COS(180 * 3.14159265359/180)
from dual

COS(180*3.14159265359/180)
--------------------------
                        -1
```

Date Arithmetic

When arithmetic operators are used with DATE datatype it is called as Date Arithmetic.
The following are the possible arithmetic operations on DATE type data.

- Adding a number to date to get the date after the given number of days.
- Subtracting a number from a date to get the date before the given number of days.
- Subtracting two dates to get the number of days between these two dates.

The following example displays the name of the student and number of days between system date and date of joining.

```
select name, sysdate - dj from students;

NAME                            No Days
------------------------------ ---------
George Micheal                 226.29194
Micheal Douglas                225.29194
Andy Roberts                   225.29194
Malcom Marshall                220.29194
Vivan Richards                 220.29194
Chirs Evert                    222.29194
Ivan Lendal                    221.29194
George Micheal                 176.29194
Richard Marx                   140.29194
Tina Turner                    140.29194
Jody Foster                    139.29194
```

In Oracle DATE datatype stores date and time. At the time of storing a date If time is not given then it will be set to 0 hours, 0 minutes and 0 seconds (beginning of the day or 12:00 a.m.).

Default date format – DD-MON-YY – doesn't include time. However time is always stored along with date and if you want to at any time you can extract the time portion a date using TO_CHAR function, which will be discussed in the next chapter.

Note: *All comparisons between two dates include time portion also.*

Note: *The fraction portion in the result between two dates indicates the difference in time.*

The following example shows the due date for first installment (assuming first installment is to be paid within 10 days from the date of joining).

```
select name,dj, dj + 10 "Due Date" from students
```

```
NAME                           DJ        Due Date
------------------------------ --------- ---------
George Micheal                 10-JAN-01 20-JAN-01
Micheal Douglas                11-JAN-01 21-JAN-01
Andy Roberts                   11-JAN-01 21-JAN-01
Malcom Marshall                16-JAN-01 26-JAN-01
Vivan Richards                 16-JAN-01 26-JAN-01
Chirs Evert                    14-JAN-01 24-JAN-01
Ivan Lendal                    15-JAN-01 25-JAN-01
George Micheal                 01-MAR-01 11-MAR-01
Richard Marx                   06-APR-01 16-APR-01
Tina Turner                    06-APR-01 16-APR-01
Jody Foster                    07-APR-01 17-APR-01
```

The following query displays the details of the payment that were made in the last 3 days:

```
select * from payments
Where  sysdate – dp <= 3;
```

Date Functions

Date functions operate on values of DATE datatype. Except MONTHS_BETWEEN all date functions return DATE data type. The following is the list of DATE functions.

Function	Description
ADD_MONTHS(date, count)	Adds *count* number of months to *date*.
MONTHS_BETWEEN (date1, date2)	Returns number of months between *date1* and *date2*.
LAST_DAY(date)	Returns the last day of the month in which *date* is.
NEXT_DAY(date, 'day')	Gives the date of next *day* after the date, where *day* name of the week like 'Monday'.
NEW_TIME(date, 'this', 'other')	Returns time in *other* time zone for time of *this* time zone.
ROUND(date)	Rounds the date depending upon the time. If time is at or after 12 hours then date is incremented. Time is always set to beginning of the day (0:0:0).
TRUNC(date)	Same as ROUND (date) but doesn't increment date.

Table 3: DATE Functions.

Adding and subtracting months

You can add or subtract months from a date using ADD_MONTHS function. If the count is positive, that many months will be added. If count is negative that many months will be subtracted.

Adding months is the process where Oracle will give the date of next specified number of months.

In the following example, Oracle will add two months to system date:

```
SQL> Select sysdate, add_months(sysdate,2)
  2  From dual;

SYSDATE   ADD_MONTH
--------- ---------
25-AUG-01 25-OCT-01
```

If the target month doesn't have the required day then the last day of the month will be taken. In the following example, 30-SEP-2001 is returned, as 31-SEP is not available. Oracle automatically adjusts the date according to the requirement.

```
SQL>  select add_months('31-aug-2001',1) from dual;

ADD_MONTH
---------
30-SEP-01
```

The following example will show the date on which the students of completed batches will be issued certificates assuming it will take 3 months time to issue certificates.

```
SQL> select bcode, ccode, enddate, add_months(enddate,3) "Cert. Date"
  2  from  batches
  3  where  enddate is not null;

BCODE CCODE ENDDATE   Cert. Dat
----- ----- --------- ---------
b1    ora   20-FEB-01 20-MAY-01
b2    asp   05-MAR-01 05-JUN-01
b3    c     27-FEB-01 27-MAY-01
b4    xml   30-MAR-01 30-JUN-01
b5    java  10-MAY-01 10-AUG-01
```

When the second parameter – *count* - is negative, date is decremented by that many months.

```
The following example shows the date on which the admissions for running batches have started.
For a batch admissions start exactly one month before the starting date.

select bcode, ccode, stdate, add_months(stdate,-1)
from  batches where enddate is null

BCODE CCODE STDATE    ADD_MONTH
----- ----- --------- ---------
b6    vbnet 12-JUL-01 12-JUN-01
b7    ora   15-AUG-01 15-JUL-01
```

Getting months between two dates

You can obtain the number of months between two dates using MONTHS_BETWEEN function. The following query returns the number of months between starting date and ending date of all completed batches.

```
select bcode, ccode, stdate, enddate,
       months_between(enddate, stdate) "NO. Months"
from batches where enddate is not null;

BCODE CCODE STDATE    ENDDATE   NO. Months
----- ----- --------- --------- ----------
b1    ora   12-JAN-01 20-FEB-01  1.2580645
b2    asp   15-JAN-01 05-MAR-01  1.6774194
b3    c     20-JAN-01 27-FEB-01  1.2258065
b4    xml   02-MAR-01 30-MAR-01  .90322581
b5    java  05-APR-01 10-MAY-01  1.1612903
```

The fraction in the result is the number of days beyond the number of months. For example, the difference between 12-JAN-01 and 20-FEB-01 is 1.258. It means there is one month and 26% of another month, which comes to 8 days.

***Note**: The fraction is calculated based on a 31-day month and also considers time portion.*

Use TRUNC function to ignore fraction in the result of MONTHS_BETWEEN as follows:

```
select bcode, ccode, stdate, enddate,
       trunc(months_between( enddate, stdate)) "NO. Months"
from batches where enddate is not null;

BCODE CCODE STDATE    ENDDATE   NO. Months
----- ----- --------- --------- ----------
b1    ora   12-JAN-01 20-FEB-01          1
b2    asp   15-JAN-01 05-MAR-01          1
b3    c     20-JAN-01 27-FEB-01          1
b4    xml   02-MAR-01 30-MAR-01          0
b5    java  05-APR-01 10-MAY-01          1
```

***Note:** It is possible to send the return value of a function to another function.*

The following query displays the batches that we completed in the last 6 months and duration is more that 1 month.

```
select bcode, ccode
from batches
where  months_between(sysdate,enddate) <= 6
       and months_between(enddate,stdate) > 1;

BCODE CCODE
----- -----
b2    asp
b3    c
b5    java
```

LAST_DAY function

LAST_DAY function returns the date of the last day of the month of the given date. The following statement displays the last day of the current month:

```
Select  sysdate, last_day(sysdate)
From  dual;

SYSDATE   LAST_DAY(
--------- ---------
25-AUG-01 31-AUG-01
```

The following query displays the due date by which first installment of each batch is to be paid. LAST_DAY function return the last day of the month in which batch has started and if 5 is added to that then it will be 5th of the next month – due date of first installment.

```
select  bcode,ccode, stdate, last_day(stdate) + 5 "Due Date"
from batches

BCODE CCODE STDATE    Due Date
----- ----- --------- ---------
b1    ora   12-JAN-01 05-FEB-01
b2    asp   15-JAN-01 05-FEB-01
b3    c     20-JAN-01 05-FEB-01
```

```
b4    xml   02-MAR-01 05-APR-01
b5    java  05-APR-01 05-MAY-01
b6    vbnet 12-JUL-01 05-AUG-01
b7    ora   15-AUG-01 05-SEP-01
```

Similarly assuming the batches are scheduled in the last week of previous month for each batch, the following query displays the date of last week:

```
select bcode, ccode, stdate, last_day(add_months(stdate,-1)) - 7 ||
              ' to ' ||  last_day(add_months(stdate), -1)
from batches

BCODE CCODE STDATE    Schd. Week
----- ----- --------- ---------------------
b1    ora   12-JAN-01 24-DEC-00 to 31-DEC-00
b2    asp   15-JAN-01 24-DEC-00 to 31-DEC-00
b3    c     20-JAN-01 24-DEC-00 to 31-DEC-00
b4    xml   02-MAR-01 21-FEB-01 to 28-FEB-01
b5    java  05-APR-01 24-MAR-01 to 31-MAR-01
b6    vbnet 12-JUL-01 23-JUN-01 to 30-JUN-01
b7    ora   15-AUG-01 24-JUL-01 to 31-JUL-01
```

NEXT_DAY function

This function returns the date of given weekday that is greater than the given date. It takes weekday – Sunday, Monday etc. – and returns the date on which the coming weekday is falling. The return value is always grater than the given date. The following example shows when is the next Friday.

```
select  sysdate, next_day(sysdate,'Fri')  from  dual;

SYSDATE   NEXT_DAY(
--------- ---------
25-AUG-01 31-AUG-01
```

If weekday of the given date and the given weekday happen to be the same then the date of coming weekday is returned. This is because the result of this function is always greater than the given date. See the following example where though the given date - 25-AUG - is Saturday NEXT_DAY returns the next Saturday and not the same date.

```
select  sysdate, next_day(sysdate,'Sat')  from  dual

SYSDATE   NEXT_DAY(
--------- ---------
25-AUG-01 01-SEP-01
```

But what if you want to get the same date, when day of the week of the given date is same as the one asked for? The following query will return the same date if date happens to fall on the required weekday.

```
select  sysdate, next_day(sysdate - 1,'Sat')  from  dual

SYSDATE   NEXT_DAY(
--------- ---------
25-AUG-01 25-AUG-01
```

When one is subtracted from the system date, though the date happens to be on Saturday it become Friday (24-AUG) because of the subtraction. Then NEXT_DAY returns the 25-AUG as it is grater than the given date – 24-AUG.

ROUND & TRUNC functions with dates

DATE data type contains both date and time. ROUND and TRUNC function can be used to round or truncate the date based on the time portion. The following query displays the date and time portion of system date using TO_CHAR function. It is suffice to know that TO_CHAR can be used to convert given date to character type using the given format.

```
Select to_char(sysdate,'dd-mm-yyyy hh24:mi:ss') from dual

TO_CHAR(SYSDATE,'DD
-------------------
25-08-2001 18:42:08
```

***Note**: TO_CHAR converts a DATE type data to character type. Please see next chapter for more information on TO_CHAR.*

ROUND function adds one day to the date if time portion of the date is greater than or equal to 12 noon.

```
select sysdate, to_char(round(sysdate),'dd-mm-yyyy hh24:mi:ss') "Round Date" from dual

SYSDATE   Round Date
--------- -------------------
25-AUG-01 26-08-2001 00:00:00
```

In the above query first ROUND function is used to round SYSDATE. As we have seen the time in SYSDATE is 18 hours, the date is incremented by one – 26-AUG. ROUND always sets the time portion to 0:0:0 (as seen in the output of the query).

TRUNC function doesn't increment the date based on the time, but it sets time portion to 0 hours, 0 minutes and 0 seconds. The following query shows the result of TRUNC function.

```
select to_char(sysdate,'dd-mm-yyyy hh24:mi:ss') "Today",
       to_char( trunc(sysdate),'dd-mm-yyyy hh24:mi:ss')  "Truncated Date"
from dual

Today               Truncated Date
------------------- -------------------
25-08-2001 18:56:53 25-08-2001 00:00:00
```

***Note**: Both ROUND and TRUNC set the time portion in the DATE data type to 12 A.M. (00:00:00).*

The following query is used to displays the details of the payments made today.

```
select * from payments
where  dp = sysdate;
```

The above query always returns *no rows selected*. This is because when Oracle compares two dates it takes date and time portions into account. In the above query, though PAYMENTS table contains rows where DP is containing the same date as SYSDATE, the time portions will not match. The remedy is to ignore time portions and compare only date portions of the dates. The following is revised query where we truncate the dates to set both the times to 0:0:0 so that only dates are compared as times are equal.

```
select * from payments
where  trunc(dp) = trunc(sysdate);
```

The following is another example where TRUNC is used to ignore time portion of DP and compare date with 10-apr-2001.

```
select * from payments
where  trunc(dp) = '10-apr-2001'

   ROLLNO DP           AMOUNT
--------- --------- ---------
       10 10-APR-01      4500
       11 10-APR-01      3500
```

Getting time in different time zone

NEW_TIME is used to return the time of the specified time zone for the time of the given time zone.

The following query displays what will be the time in PST for the time in GMT.

```
select to_char(sysdate,'dd-mm-yyyy hh24:mi:ss')  GMT ,
         to_char(  new_time(sysdate,'GMT','AST'),'dd-mm-yyyy hh24:mi:ss')  AST
from dual

TO_CHAR(SYSDATE,'DD TO_CHAR(NEW_TIME(SY
------------------- -------------------
25-08-2001 19:35:36 25-08-2001 15:35:36
```

The following are a few of the available time zones. For complete lit, please see Oracle online documentation.

Time Zone	Meaning
EST , EDT	Eastern Standard or Daylight Time
GMT	Greenwich Mean Time
HST , HDT	Alaska-Hawaii Standard Time or Daylight Time.
PST , PDT	Pacific Standard or Daylight Time
AST , ADT	Atlantic Standard or Daylight Time

Summary

A function is to perform a single operation and return a value. Functions are of different types. Arithmetic functions are used to perform arithmetic operations. Date functions perform operations on date type data. Performing arithmetic operations on dates is called as *date arithmetic.* DUAL table is with SELECT command to display the result of expression that do not relate to any table.

Exercises

1. _____ function can be used to subtract months from a date.
2. The return value of ROUND (2323.343,2) is _____.
3. To get the remainder of a division_____ function is used.
4. In Date Arithmetic _____, _____ and _____ operations are allowed.
5. _____ is the result of LAST_DAY(SYSDATE) assuming SYSDATE is 24th August.
6. Which function can be used to set time portion of the DATE data type to 00:00:00, without affecting date portion.? _____
7. Display details of students who have joined in the last 4 months.
8. Display ROLLNO, NAME, DJ and number of days between current date and DJ for each student.
9. Display the first Sunday since batch with code 2 started.
10. Display details of batches that started three or more months back.
11. Display the details of payments of last Monday.
12. _____ is the function to get number of years between two dates.