Как правило, при работе с разными типами данных **PL/SQL** сам неявно вызывает функции преобразования и вам не приходиться самим что-то делать. Но **PL/SQL** использует их, с типом преобразования по умолчанию. Например, запишем такой блок:

```
SET SERVEROUTPUT ON

BEGIN

DBMS_OUTPUT.put_line(SYSDATE);

END;
/
```

Получаем:

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2
  3  DBMS_OUTPUT.put_line(SYSDATE);
  4
  5  END;
  6  /
16.11.03

Процедура PL/SQL успешно завершена.
```

Такой, тип вывода нас не всегда устраивает, по этому у функции **TO_CHAR** есть множество форматов вывода данных, которые можно применять используя явное преобразование или, проще говоря, ее вызов. Функция **TO_CHAR** имеет следующий синтаксис:

```
TO_CHAR(d, [, формат [nls_параметр]])
```

Где **d**, это данные типа **DATE**, а "формат" - это строка, которая состоит из нескольких или одного элементов, описанных ниже в таблице. При этом, если "формат" не указан, то как вы помните, используется формат по умолчанию для конкретного сеанса. "nls_параметр" - управляет выбором языка, но, как правило, достаточно языка по умолчанию, например я этот параметр использовал очень редко. Далее можете посмотреть на таблицу форматов и пример с применением данных из таблицы.

Элемент формата дат	Описание
Знак пунктуации	Все знаки пунктуации дублируются в результирующей строке символов
"Текст"	Текст, заключенный в двойные кавычки так же дублируется
AD, A.D.	Показатель "нашей эры" (с точками или без точек.)
AM, A.M.	Показатель времени до полудня (с точками или без точек.)
BC, B.C.	Показатель "до нашей эры" (с точками или без точек.)
CC, SCC	Век SCC возвращает даты "до нашей эры" как отрицательные значения
D	День недели (1-7)
DAY	Название дня, дополненное пробелами до девяти символов. *
DD	День месяца (1 - 31)
DDD	День года (1 - 366)

DY	Сокращенное название дня. *
IW	Неделя года (1 - 52), (1 - 53) (в основе лежит стандарт ISO)
IYY, IY, I	Последние три, две или одна цифра года ISO
IYYY	Четырех цифровое обозначение года, основанное на стандарте ISO
HH, HH12	Час дня (1-12)
HH24	Час дня (0-23)
J	День по Юлианскому календарю. Число дней с 1 января 4712 года до н.э. Соответствующий результат будет целым значением.
MI	Минута (0-59)
MM	Месяц (1 - 12), JAN = 1, DEC = 12
MONTH	Название месяца, дополненное пробелами до девяти символов. *
MON	Сокращенное название месяца. *
PM P.M.	Показатель времени после полудня (с точками или без точек.)
Q	Квартал года (1 - 4) С января по март - первый квартал.
RM	Месяц, обозначенный римскими цифрами. (I - XII) JAN = I, DEC = XII
RR	Последние две цифры года для других веков.
SS	Секунды (0-59)
SSSS	Секунды после полуночи (0 - 86399) Модель формата 'J.SSSSS' всегда будет давать в результате числовое значение.
WW	Неделя года (1-53). Неделя 1 начинается с первого дня года и продолжается до седьмого дня. таким образом, недели не всегда начинаются с воскресенья как это принято в США, это что то вроде сквозного недельного отсчета.
W	Неделя месяца (1-5) Недели определяются, так же как и WW!
Y, YYY	Год с запятой в указанной позиции.
YEAR, SYEAR	Год, записанный буквами, SYEAR возвращает даты до нашей эры как отрицательные значения. *
YYYY, SYYYY	Четырех цифровой год. Возвращает даты до нашей эры как отрицательные значения.
YYY, YY, Y	Последние три, две или одна цифра года.

Вот блок примера, где я постарался применить разнообразные форматы вывода функции **TO_CHAR()**. Для получения данных с типом **DATE** я применил функцию **SYSDATE**, с которой вы уже знакомы:

```
SET SERVEROUTPUT ON

BEGIN

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MM-YY'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MM-YYYY A.D.'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-J-MM-YYYY A.D.'));

        -- Function  TO_CHAR() --
```

```
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'D-DAY-DD-MON, YYYY'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DAY DDD MONTH YYYY'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DAY-MONTH-YYYY'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'Q-DD-RM-YYYY '));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MONTH-WW-YYYY HH24:MI:SS'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MONTH-W-YYYY HH24:MI:SS'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MONTH-YEAR HH24:MI:SS'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH24:MI:SS'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH:MI:SS AM'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH12:MI:SS P.M.'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH12:MI:SS:J.SSSSS P.M.'));

END;
/
```

После **SQL*Plus** получаем:

```
SQL> BEGIN
  2
  3     -- Function  TO_CHAR() --
  4     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MM-YY'));
  5
  6     -- Function  TO_CHAR() --
  7     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MM-YYYY A.D.'));
  8
  9     -- Function  TO_CHAR() --
 10     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-J-MM-YYYY A.D.'));
 11
 12     -- Function  TO_CHAR() --
 13     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'D-DAY-DD-MON, YYYY'));
 14
 15     -- Function  TO_CHAR() --
 16     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DAY DDD MONTH YYYY'));
 17
 18     -- Function  TO_CHAR() --
 19     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DAY-MONTH-YYYY'));
 20
 21     -- Function  TO_CHAR() --
 22     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'Q-DD-RM-YYYY '));
 23
 24     -- Function  TO_CHAR() --
 25     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MONTH-WW-YYYY HH24:MI:SS'));
 26
 27     -- Function  TO_CHAR() --
```

```
 28     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MONTH-W-YYYY HH24:MI:SS'));
 29
 30     -- Function  TO_CHAR() --
 31     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'DD-MONTH-YEAR HH24:MI:SS'));
 32
 33     -- Function  TO_CHAR() --
 34     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH24:MI:SS'));
 35
 36     -- Function  TO_CHAR() --
 37     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH:MI:SS AM'));
 38
 39     -- Function  TO_CHAR() --
 40     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH12:MI:SS P.M.'));
 41
 42     -- Function  TO_CHAR() --
 43     DBMS_OUTPUT.put_line( TO_CHAR(SYSDATE,  'HH12:MI:SS:J.SSSSS P.M.'));
 44
 45  END;
 46  /
16-11-03
16-11-2003 Н.Э.
16-2452960-11-2003 Н.Э.
7-ВОСКРЕСЕНЬЕ-16-НОЯ, 2003
ВОСКРЕСЕНЬЕ 320 НОЯБРЬ   2003
ВОСКРЕСЕНЬЕ-НОЯБРЬ  -2003
4-16-XI  -2003
16-НОЯБРЬ  -46-2003 13:13:14
16-НОЯБРЬ  -3-2003 13:13:14
16-НОЯБРЬ  -TWO THOUSAND THREE 13:13:14
13:13:14
01:13:14 PM
01:13:14 PM
01:13:14:2452960.47594 PM

Процедура PL/SQL успешно завершена.
```

Вот таким образом можно, применяя разнообразные форматы вывода, получать необходимую для работы структуру данных.

Для некоторых форматов при преобразовании дат есть чувствительность к регистру (они в таблице имеют значок *). Например, функция **MON** вернет **JAN**, а если записать **Mon**, то получите **Jan**!

Итак, определим такой блок:

```
SET SERVEROUTPUT ON

BEGIN

        -- Function  ADD_MONTHS() --
        DBMS_OUTPUT.put_line('46*-> '||ADD_MONTHS('09-02-2000', 5));

        -- Function  ADD_MONTHS() --
        DBMS_OUTPUT.put_line('47*-> '||ADD_MONTHS('18-06-2001', 3));

        -- Function  ADD_MONTHS() --
        DBMS_OUTPUT.put_line('48*-> '||ADD_MONTHS('12-02-1981', 2));

        -- Function  LAST_DAY() --
        DBMS_OUTPUT.put_line('49*-> '||LAST_DAY('09-02-2000')||' '||LAST_DAY('19-02-
2001'));

        -- Function  LAST_DAY() --
        DBMS_OUTPUT.put_line('50*-> '||LAST_DAY('05-03-2002'));
```

```
        -- Function  MONTHS_BETWEEN() --
        DBMS_OUTPUT.put_line('51*-> '||MONTHS_BETWEEN('16-04-1973', '16-03-1997'));

        -- Function  MONTHS_BETWEEN() --
        DBMS_OUTPUT.put_line('52*-> '||MONTHS_BETWEEN('18-04-1972', '23-03-1961'));

        -- Function  NEW_TIME() --
        DBMS_OUTPUT.put_line( '53*-> '||TO_CHAR(NEW_TIME(TO_DATE('12-04-1971 12:00:00',
'DD-MM-YYYY HH24:MI:SS'),
                'PST', 'EST'), 'DD-MON-YYYY HH24:MI:SS')||': Pacific -> Eastern' );

        -- Function  NEXT_DAY() --
        DBMS_OUTPUT.put_line( '54*-> '||NEXT_DAY('07-04-2003', 'Понедельник'));

        -- Function  NEXT_DAY() --
        DBMS_OUTPUT.put_line( '55*-> '||NEXT_DAY('01-03-1998', 'Четверг'));

END;
/
```

После запуска в **SQL*Plus** получаем:

```
SQL> BEGIN
  2
  3     -- Function  ADD_MONTHS() --
  4     DBMS_OUTPUT.put_line('46*-> '||ADD_MONTHS('09-02-2000', 5));
  5
  6     -- Function  ADD_MONTHS() --
  7     DBMS_OUTPUT.put_line('47*-> '||ADD_MONTHS('18-06-2001', 3));
  8
  9     -- Function  ADD_MONTHS() --
 10     DBMS_OUTPUT.put_line('48*-> '||ADD_MONTHS('12-02-1981', 2));
 11
 12     -- Function  LAST_DAY() --
 13     DBMS_OUTPUT.put_line('49*-> '||LAST_DAY('09-02-2000')||' '||LAST_DAY('19-02-
2001'));
 14
 15     -- Function  LAST_DAY() --
 16     DBMS_OUTPUT.put_line('50*-> '||LAST_DAY('05-03-2002'));
 17
 18     -- Function  MONTHS_BETWEEN() --
 19     DBMS_OUTPUT.put_line('51*-> '||MONTHS_BETWEEN('16-04-1973', '16-03-1997'));
 20
 21     -- Function  MONTHS_BETWEEN() --
 22     DBMS_OUTPUT.put_line('52*-> '||MONTHS_BETWEEN('18-04-1972', '23-03-1961'));
 23
 24     -- Function  NEW_TIME() --
 25     DBMS_OUTPUT.put_line( '53*-> '||TO_CHAR(NEW_TIME(TO_DATE('12-04-1971 12:00:00',
'DD-MM-YYYY HH24:MI:SS'),
 26             'PST', 'EST'), 'DD-MON-YYYY HH24:MI:SS')||': Pacific -> Eastern' );
 27
 28     -- Function  NEXT_DAY() --
 29     DBMS_OUTPUT.put_line( '54*-> '||NEXT_DAY('07-04-2003', 'Понедельник'));
 30
 31     -- Function  NEXT_DAY() --
 32     DBMS_OUTPUT.put_line( '55*-> '||NEXT_DAY('01-03-1998', 'Четверг'));
 33
 34  END;
 35  /
46*-> 09.07.00
47*-> 18.09.01
48*-> 12.04.81
49*-> 29.02.00 28.02.01
```

```
50*-> 31.03.02
51*-> -287
52*-> 132,838709677419354838709677419354838 71
53*-> 12-АПР-1971 15:00:00: Pacific -> Eastern
54*-> 14.04.03
55*-> 05.03.98

Процедура PL/SQL успешно завершена.
```

Если у кого-то, данный блок вызвал ошибки ничего страшного нет, просто у вас скорее всего не выставлен язык системы, об этом мы уже говорили. А, так же формат даты в самой **Windows** установите **dd.MM.YYYY**.

Теперь давайте рассмотрим особенности работы с функцией **TO_CHAR()** применительно к преобразованию, чисел. При выполнении этой операции функция имеет следующий синтаксис:

```
TO_CHAR(число, [, формат[, nls_параметр]])
```

При этом функция **TO_CHAR()** преобразует тип **NUMBER** в тип **VARCHAR2**. В таблице указано, какие есть форматы и как они используются при преобразовании:

Элемент формата	**Пример строки данного формата**	**Описание**
9	99	Каждая цифра 9 представляет значащую цифру результата. Число значащих цифр возвращаемого значения равно числу цифр 9, отрицательное значение предваряется знаком минуса. Все начальные нули заменяются пробелами.
0	0999	Возвращается число с начальными нулями, а не пробелами.
0	9990	Возвращается число с конечными нулями, а не пробелами.
$	$999	Возвращаемое значение предваряется знаком доллара не зависимо от используемого символа денежной единицы можно применить совместно с начальными или конечными нулями.
B	B999	Вместо нулевой целой части десятичного число возвращаются пробелы.
MI	999MI	Возвращает отрицательное число, у которого знак минуса указан не в начале, а в конце. В положительном значении на этом месте будет пробел.
S	S9999	Возвращаемое число предваряется знаком: + для положительных, чисел - для отрицательных.
S	9999S	Возвращаемое число заканчивается знаком: + для положительных, чисел - для отрицательных.
PR	99PR	Возвращается отрицательное число в угловых скобках "<", ">". У положительных чисел, на этом месте пробелы.
D	99D9	Возвращает число с десятичной точкой в указанной позиции. Число 9 с обеих сторон указывает максимальное число цифр.
G	9G999	Возвращает число с разделителем групп в указанной позиции. **G** может появляться в указанной строке формата неоднократно.

C	C99	Возвращает число с символом денежной единицы ISO в указанной позиции. **C** может появляться в указанной строке формата неоднократно.
L	L999	Возвращает число с символом денежной единицы национального языка в указанной позиции.
,	999,999	Возвращает число с запятой в указанной позиции, не зависимо от выбранного разделителя групп.
.	99.99	Возвращает число с десятичной точкой в указанной позиции, не зависимо от выбранного десятичного разделителя.
V	99V999	Возвращает число, умноженное на **10n**, где **n** - это число цифр 9 после V. При необходимости значение округляется.
EEEE	9.99EEEE	Возвращает число в экспоненциальном представлении.
RM	RM	Возвращает число при помощи римских цифр верхнего регистра.

Для примера, приведем вот такой блок:

```
SET SERVEROUTPUT ON

BEGIN

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(534523));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(34387,  '99999'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(5000,  '$9999'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(-9,  '9S'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(-34,  'S99'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(3 - 5,  '999MI'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(7 - 3,  'S9'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(4 - 5,  '99PR'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(8900,  'L9999'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(10000000,  '9.9EEEE'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(10,  'RM'));

        -- Function  TO_CHAR() --
        DBMS_OUTPUT.put_line( TO_CHAR(105,  'RM'));

END;
/
```

После запуска получаем:

```
SQL> BEGIN
  2
  3     -- Function  TO_CHAR() --
  4     DBMS_OUTPUT.put_line( TO_CHAR(534523));
  5
  6     -- Function  TO_CHAR() --
  7     DBMS_OUTPUT.put_line( TO_CHAR(34387,  '99999'));
  8
  9     -- Function  TO_CHAR() --
 10     DBMS_OUTPUT.put_line( TO_CHAR(5000,  '$9999'));
 11
 12     -- Function  TO_CHAR() --
 13     DBMS_OUTPUT.put_line( TO_CHAR(-9,  '9S'));
 14
 15     -- Function  TO_CHAR() --
 16     DBMS_OUTPUT.put_line( TO_CHAR(-34,  'S99'));
 17
 18     -- Function  TO_CHAR() --
 19     DBMS_OUTPUT.put_line( TO_CHAR(3 - 5,  '999MI'));
 20
 21     -- Function  TO_CHAR() --
 22     DBMS_OUTPUT.put_line( TO_CHAR(7 - 3,  'S9'));
 23
 24     -- Function  TO_CHAR() --
 25     DBMS_OUTPUT.put_line( TO_CHAR(4 - 5,  '99PR'));
 26
 27     -- Function  TO_CHAR() --
 28     DBMS_OUTPUT.put_line( TO_CHAR(8900,  'L9999'));
 29
 30     -- Function  TO_CHAR() --
 31     DBMS_OUTPUT.put_line( TO_CHAR(10000000,  '9.9EEEE'));
 32
 33     -- Function  TO_CHAR() --
 34     DBMS_OUTPUT.put_line( TO_CHAR(10,  'RM'));
 35
 36     -- Function  TO_CHAR() --
 37     DBMS_OUTPUT.put_line( TO_CHAR(105,  'RM'));
 38
 39  END;
 40  /
534523
34387
$5000
9-
-34
2-
+4
<1>
p.8900
1.0E+07
X
CV

Процедура PL/SQL успешно завершена.
```

И для полноты картины, давайте рассмотрим функцию **TO_NUMBER()**, так как с **TO_CHAR()** они практически близнецы, но с двух противоположных сторон. Синтаксис у **TO_NUMBER** следующий:

```
TO_NUMBER(строка_символов, [, формат [, nls_параметр]])
```

В данном случае "строка_символов" - это тип **CHAR** или **VARCHAR2**, которая применительно к "формат", преобразуется в типу **NUMBER**. "формат" в данной функции, тот же что и для **TO_CHAR()**. Вот так достаточно просто. А, если в предыдущем примере поменять, например, третью снизу строку вот так:

```
DBMS_OUTPUT.put_line( TO_CHAR(TO_NUMBER('10000000',  '9.9EEEE'), '9.9EEEE') );
```

РАБОТА СО ВСТРОЕНЫМИ ФУНКЦИЯМИ

```
SET SERVEROUTPUT ON

BEGIN

        DBMS_OUTPUT.enable;
        -- Function CHR() --
        DBMS_OUTPUT.put_line('1*->'||CHR(37)||' '||CHR(38)||' '||CHR(101)||'
'||CHR(105));

        -- Function CONCAT() --
        DBMS_OUTPUT.put_line('2*->'||CONCAT('Vasiya', 'Pupkin'));

        -- Function INITCAP() --
        DBMS_OUTPUT.put_line('3*->'||INITCAP('iF yoU or mY scoRE and 7 YEARS ago
...'));

        -- Function LOWER() --
        DBMS_OUTPUT.put_line('4*->'||LOWER('iF yoU or mY scoRE and 7 YEARS ago ...'));

        -- Function LPAD() 1 --
        DBMS_OUTPUT.put_line('5*->'||LPAD('Short String', 20, 'XY'));

        -- Function LPAD() 2 --
        DBMS_OUTPUT.put_line('6*->'||LPAD('Short String', 13, 'PD'));

        -- Function LTRIM() 1 --
        DBMS_OUTPUT.put_line('7*->'||LTRIM('     The White House has a many tree!'));

        -- Function LTRIM() 2 --
        DBMS_OUTPUT.put_line('8*->'||LTRIM('xxxxxThe White House has a many tree!',
'x'));

        -- Function LTRIM() 3 --
        DBMS_OUTPUT.put_line('9*->'||LTRIM('xyxyxyxyxyThe White House has a many
tree!', 'xy'));

        -- Function LTRIM() 4 --
        DBMS_OUTPUT.put_line('10*->'||LTRIM('xyxyxxxxxyThe White House has a many
tree!', 'xy'));

        -- Function REPLACE() 1 --
        DBMS_OUTPUT.put_line('11*->'||REPLACE('This and That', 'Th', 'B'));

        -- Function REPLACE() 2 --
        DBMS_OUTPUT.put_line('12*->'||REPLACE('This and That', 'Th'));

        -- Function REPLACE() 3 --
        DBMS_OUTPUT.put_line('13*->'||REPLACE('This and That', NULL));

END;
/
```

После запуска получаем:

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2
  3   DBMS_OUTPUT.enable;
  4   -- Function CHR() --
  5   DBMS_OUTPUT.put_line('1*->'||CHR(37)||' '||CHR(38)||' '||CHR(101)||'
'||CHR(105));
  6
  7   -- Function CONCAT() --
  8   DBMS_OUTPUT.put_line('2*->'||CONCAT('Vasiya', 'Pupkin'));
  9
 10   -- Function INITCAP() --
 11   DBMS_OUTPUT.put_line('3*->'||INITCAP('iF yoU or mY scoRE and 7 YEARS ago ...'));
 12
 13   -- Function LOWER() --
 14   DBMS_OUTPUT.put_line('4*->'||LOWER('iF yoU or mY scoRE and 7 YEARS ago ...'));
 15
 16   -- Function LPAD() 1 --
 17   DBMS_OUTPUT.put_line('5*->'||LPAD('Short String', 20, 'XY'));
 18
 19   -- Function LPAD() 2 --
 20   DBMS_OUTPUT.put_line('6*->'||LPAD('Short String', 13, 'PD'));
 21
 22   -- Function LTRIM() 1 --
 23   DBMS_OUTPUT.put_line('7*->'||LTRIM('     The White House has a many tree!'));
 24
 25   -- Function LTRIM() 2 --
 26   DBMS_OUTPUT.put_line('8*->'||LTRIM('xxxxxThe White House has a many tree!',
'x'));
 27
 28   -- Function LTRIM() 3 --
 29   DBMS_OUTPUT.put_line('9*->'||LTRIM('xyxyxyxyxyThe White House has a many tree!',
'xy'));
 30
 31   -- Function LTRIM() 4 --
 32   DBMS_OUTPUT.put_line('10*->'||LTRIM('xyxyxxxxxyThe White House has a many tree!',
'xy'));
 33
 34   -- Function REPLACE() 1 --
 35   DBMS_OUTPUT.put_line('11*->'||REPLACE('This and That', 'Th', 'B'));
 36
 37   -- Function REPLACE() 2 --
 38   DBMS_OUTPUT.put_line('12*->'||REPLACE('This and That', 'Th'));
 39
 40   -- Function REPLACE() 3 --
 41   DBMS_OUTPUT.put_line('13*->'||REPLACE('This and That', NULL));
 42
 43  END;
 44  /
1*->% & e i
2*->VasiyaPupkin
3*->If You Or My Score And 7 Years Ago ...
4*->if you or my score and 7 years ago ...
5*->XYXYXYXYShort String
6*->PShort String
7*->The White House has a many tree!
8*->The White House has a many tree!
9*->The White House has a many tree!
10*->The White House has a many tree!
11*->Bis and Bat
12*->is and at
13*->This and That
```

```
Процедура PL/SQL успешно завершена.
```

Все это относится к СИМВОЛЬНЫМ ФУНКЦИЯМ, ВОЗВРАЩАЮЩИМ СИМВОЛЬНЫЕ ЗНАЧЕНИЯ.

CHR(x)
Возвращает символ, имеющий код, равный x в наборе символов БД. Пример строка 1*.

CONCAT(строка 1, строка2)
Возвращает "строка 1", конкатенированную, (сцепленную) со "строка 2". То же что и операция "||"! Пример строка 2*.

INITCAP(строка)
Возвращает "строка", в которой каждое слово начинается с прописной буквы и продолжается строчными. Слова разделяются пробелами или не буквенно-цифровыми символами. Символы не являющиеся буквами не изменяются. Пример строка 3*.

LOWER(строка)
Возвращает "строка", со строчными символами. Символы не являющиеся буквами не изменяются. Пример строка 4*.

LPAD(строка 1, x, строка 2)
Вот интересная функция! :) Возвращает "строка 1", дополненную слева до размера **x** символами "строка 2". Если размер "строка 2", меньше **x**, то при необходимости она дублируется. Если размер "строка 2" больше **x**, то берутся только первые x ее символов. Если "строка 2" не указана, то ее заменяют символы пробела. Пример строка 5*, 6*.

LTRIM(строка 1, строка 2)
Возвращает "строка 1", в которой удалены крайние левые символы, идентичные символам "строка 2". Значением по умолчанию для "строка 2", является знак пробела. "строка 1" просматривается с левого края, и при встрече первого символа не совпадающего с "строка 2", возвращается результат. Пример строка 7*, 8*, 9*, 10*.

REPLACE(строка_символов, строка_поиска, [строка_замены])
Возвращает "строка_символов", в которой каждое вхождение "строка_поиска", заменяется на "строка_замены". Если "строка_замены", не указана, то все вхождения "строка_поиска", удаляются из "строка_символов". Пример строка 11*, 12*, 13*.

Далее, продолжим.

Продолжим разбор функций **PL/SQL**. Символьные функции далее:

```
SET SERVEROUTPUT ON

BEGIN

        -- Function RPAD() 1 --
        DBMS_OUTPUT.put_line('14*->'||RPAD('This is Cool', 15, '!'));

        -- Function RPAD() 2 --
        DBMS_OUTPUT.put_line('15*->'||RPAD('This is Cool', 19, 'MA'));

        -- Function RTRIM() 1 --
        DBMS_OUTPUT.put_line('16*->'||RTRIM('The White House has a many tree!
'));

        -- Function RTRIM() 2 --
        DBMS_OUTPUT.put_line('17*->'||RTRIM('The White House has a many tree!xxxxxx',
'x'));

        -- Function RTRIM() 3 --
```

```
        DBMS_OUTPUT.put_line('18*->'||RTRIM('The White House has a many
tree!xyxyxyxyxy', 'xy'));

        -- Function RTRIM() 4 --
        DBMS_OUTPUT.put_line('19*->'||RTRIM('The White House has a many
tree!xyxyxxxxxy', 'xy'));

        -- Function SUBSTR() 1 --
        DBMS_OUTPUT.put_line('20*->'||SUBSTR('Braun Fox',1,5));

        -- Function SUBSTR() 2 --
        DBMS_OUTPUT.put_line('21*->'||SUBSTR('Braun Fox',-3,3));

        -- Function SUBSTR() 3 --
        DBMS_OUTPUT.put_line('22*->'||SUBSTR('Braun Fox',3));

        -- Function TRANSLATE() 1 --
        DBMS_OUTPUT.put_line('23*->'||TRANSLATE('KrotPups', 'Pups', 'Boss'));

        -- Function TRANSLATE() 2 --
        DBMS_OUTPUT.put_line('24*->'||TRANSLATE('KrotPups', 'KrotPups', 'Kolobok'));

        -- Function UPPER() --
        DBMS_OUTPUT.put_line('25*->'||UPPER('THE quick bROwn Fox jumped oVer THE LaZy
dOG ... '));

END;
/

SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2
  3   -- Function RPAD() 1 --
  4   DBMS_OUTPUT.put_line('14*->'||RPAD('This is Cool', 15, '!'));
  5
  6   -- Function RPAD() 2 --
  7   DBMS_OUTPUT.put_line('15*->'||RPAD('This is Cool', 19, 'MA'));
  8
  9   -- Function RTRIM() 1 --
 10   DBMS_OUTPUT.put_line('16*->'||RTRIM('The White House has a many tree!       '));
 11
 12   -- Function RTRIM() 2 --
 13   DBMS_OUTPUT.put_line('17*->'||RTRIM('The White House has a many tree!xxxxxx',
'x'));
 14
 15   -- Function RTRIM() 3 --
 16   DBMS_OUTPUT.put_line('18*->'||RTRIM('The White House has a many tree!xyxyxyxyxy',
'xy'));
 17
 18   -- Function RTRIM() 4 --
 19   DBMS_OUTPUT.put_line('19*->'||RTRIM('The White House has a many tree!xyxyxxxxxy',
'xy'));
 20
 21   -- Function SUBSTR() 1 --
 22   DBMS_OUTPUT.put_line('20*->'||SUBSTR('Braun Fox',1,5));
 23
 24   -- Function SUBSTR() 2 --
 25   DBMS_OUTPUT.put_line('21*->'||SUBSTR('Braun Fox',-3,3));
 26
 27   -- Function SUBSTR() 3 --
 28   DBMS_OUTPUT.put_line('22*->'||SUBSTR('Braun Fox',3));
 29
 30   -- Function TRANSLATE() 1 --
 31   DBMS_OUTPUT.put_line('23*->'||TRANSLATE('KrotPups', 'Pups', 'Boss'));
```

```
 32
 33   -- Function TRANSLATE() 2 --
 34   DBMS_OUTPUT.put_line('24*->'||TRANSLATE('KrotPups', 'KrotPups', 'Kolobok'));
 35
 36   -- Function UPPER() --
 37   DBMS_OUTPUT.put_line('25*->'||UPPER('THE quick bROwn Fox jumped oVer THE LaZy dOG
... '));
 38
 39  END;
 40  /
14*->This is Cool!!!
15*->This is CoolMAMAMAM
16*->The White House has a many tree!
17*->The White House has a many tree!
18*->The White House has a many tree!
19*->The White House has a many tree!
20*->Braun
21*->Fox
22*->aun Fox
23*->KrotBoss
24*->Kolobok
25*->THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG ...

Процедура PL/SQL успешно завершена.
```

RPAD(строка 1, x, строка 2)

Вот еще интересная функция! :) Возвращает "строка 1", дополненную справа до размера **x** символами "строка 2". Если размер "строка 2" меньше **x**, то при необходимости она дублируется. Если размер "строка 2" больше **x** то берутся только первые **x** ее символов. Если "строка 2" не указана, то ее заменяют символы пробела. Пример, строка 5*, 6*. Обратите внимание, что x указывается как размер строки символов, отображаемой на экране, а не как реальный размер. Пример 14*, 15*.

RTRIM(строка 1, строка 2)

Возвращает "строка 1", в которой удалены крайние правые символы, идентичные символам "строка 2". Значением по умолчанию для "строка 2", является знак пробела. "строка 1" просматривается с левого края, и при встрече первого символа не совпадающего с "строка 2", возвращается результат. Пример строка 16*, 17*, 18*, 19*.

SUBSTR(строка 1, a, [,b])

Вот с этой функцией, я наиболее часто работал, очень удобная штучка! Возвращает часть "строка 1", начинающуюся с символа с номером **a**, и имеющую длину **b** символов. Если **a = 0**, это равносильно тому, что **a = 1** (начало строки) если **b** положительно возвращаются символы слева направо. Если **b** отрицательно то, начиная с конца строки и считаются справа налево! Если **b** отсутствует, то по умолчанию возвращаются все символы, до конца строки. Если **b** меньше 1, то возвращается значение **NULL**. Если в качестве **a** и **b**, указано число с плавающей точкой, его дробная часть отбрасывается! Пример строка 20*, 21*, 22*.

TRANSLATE(строка_символов, заменяемая_трока, вносимая_строка)

Возвращает "строка_символов", в которой все вхождения каждого символа "заменяемая_трока" замещаются соответствующим символом "вносимая_строка". Функция **TRANSLATE**, является расширением функции **REPLACE**. Если "заменяемая_трока" длиннее чем "вносимая_строка", все ее лишние символы удаляются поскольку для них нет соответствующих символов во "вносимая_строка". "вносимая_строка" не может быть пустой. **Oracle** интерпретирует пустую строку как значение **NULL**, а если любой аргумент функции **TRANSLATE** является **NULL**, то результат тоже будет **NULL**. Пример строка 23*, 24*.

UPPER(строка)

Возвращает "строка", в которой все символы прописные. Символы не являющиеся буквами не изменяются. Пример, строка 4*. Пример строка 25*.

Вот, с символьными функциями возвращающими символьные значения пока все. Далее продолжим, следующую группу функций **PL/SQL**.

```
SET SERVEROUTPUT ON

BEGIN

        -- Function  ASCII() --
        DBMS_OUTPUT.put_line('26*-> '||TO_CHAR(ASCII('A'))||' '||TO_CHAR(ASCII(' '))||'
'||TO_CHAR(ASCII('F')));

        -- Function  INSTR() --
        DBMS_OUTPUT.put_line('27*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il', 1,
2)));

        -- Function  INSTR() --
        DBMS_OUTPUT.put_line('28*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il', -1,
2)));

        -- Function  INSTR() --
        DBMS_OUTPUT.put_line('29*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il', 8)));

        -- Function  INSTR() --
        DBMS_OUTPUT.put_line('30*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il',
18)));

        -- Function  LENGTH() --
        DBMS_OUTPUT.put_line('31*-> '||TO_CHAR(LENGTH('Hello World!!!')));

        -- Function  LENGTH() --
        DBMS_OUTPUT.put_line('32*-> '||TO_CHAR(LENGTH('Miller is very ill!')));

        DBMS_OUTPUT.put_line('Part2 Number function.');

        -- Function  ABS() --
        DBMS_OUTPUT.put_line('33*-> '||TO_CHAR(ABS(-5))||' '||TO_CHAR(ABS(5)));

        -- Function  ACOS() ASIN() --
        DBMS_OUTPUT.put_line('34*-> '||TO_CHAR(ACOS(0.5))||' '||TO_CHAR(ASIN(0.7)));

END;
/
```

После запуска, смотрим результат:

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2
  3   -- Function  ASCII() --
  4   DBMS_OUTPUT.put_line('26*-> '||TO_CHAR(ASCII('A'))||' '||TO_CHAR(ASCII(' '))||'
'||TO_CHAR(ASCII('F')));
  5
  6   -- Function  INSTR() --
  7   DBMS_OUTPUT.put_line('27*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il', 1,
2)));
  8
  9   -- Function  INSTR() --
 10   DBMS_OUTPUT.put_line('28*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il', -1,
2)));
 11
 12   -- Function  INSTR() --
 13   DBMS_OUTPUT.put_line('29*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il', 8)));
 14
 15   -- Function  INSTR() --
 16   DBMS_OUTPUT.put_line('30*-> '||TO_CHAR(INSTR('Miller is very ill!', 'il', 18)));
 17
```

```
 18   -- Function  LENGTH() --
 19   DBMS_OUTPUT.put_line('31*-> '||TO_CHAR(LENGTH('Hello World!!!')));
 20
 21   -- Function  LENGTH() --
 22   DBMS_OUTPUT.put_line('32*-> '||TO_CHAR(LENGTH('Miller is very ill!')));
 23
 24   DBMS_OUTPUT.put_line('Part2 Number function.');
 25
 26   -- Function  ABS() --
 27   DBMS_OUTPUT.put_line('33*-> '||TO_CHAR(ABS(-5))||' '||TO_CHAR(ABS(5)));
 28
 29   -- Function  ACOS() ASIN() --
 30   DBMS_OUTPUT.put_line('34*-> '||TO_CHAR(ACOS(0.5))||' '||TO_CHAR(ASIN(0.7)));
 31
 32  END;
 33  /
26*-> 65 32 70
27*-> 16
28*-> 2
29*-> 16
30*-> 0
31*-> 14
32*-> 19
Part2 Number function.
33*-> 5 5
34*-> 1,047197551196597746154214461093167628 05
,775397496610753063740353352714987113548 8

Процедура PL/SQL успешно завершена.
```

Итак, поехали:

ASCII(строка)

Возвращает десятичное представление первого байта "строка", согласно применяемому набору символов. Пример 26*.

INSTR(строка 1, строка 2, [,a[,b]])

А, вот эта функция, просто так не разберешься! :) Но, полезная когда поймешь, как она действует. Итак! Возвращает местоположение "строка 2", в "строка 1". "строка 1" просматривается слева, начиная с позиции **a**. Если **a** отрицательно, то "строка 1", просматривается справа. Возвращается позиция указывающая местоположение **b**-го вхождения. Значением по умолчанию, как для **a** так и для **b** является 1, что дает в результате позицию, первого вхождения, "строка 2", в "строка 1". Если при заданных **a** и **b**, "строка 2" не найдена, возвращается 0. Пример 27*, 28*, 29*, 30*.

LENGTH(строка)

Очень полезная функция! :) Возвращает размер "строка" в символах. Значения типа **CHAR** дополняются пробелами. по этому если "строка" имеет тип **CHAR**, в размере указываются и конечные пробелы. Если "строка", является **NULL** - значением, то и возвращается **NULL**! Пример 31*, 32*.

С символьными функциями пока все! Переходим к числовым. Они, может и редко будут вами применяться, но знать их не помешает! К слову, эти функции в качестве аргументов используют и возвращают значения типа **NUMBER**!

ABS(x)

Возвращает абсолютное значение для **x**. Пример 33*.

ACOS(x) ASIN(x)

Возвращает арккосинус и арксинус для **x** соответственно. Обратите внимание на количество значащих после запятой! Пример 34*.

```
SET SERVEROUTPUT ON
```

```
BEGIN

        -- Function  ATAN() TAN() --
        DBMS_OUTPUT.put_line('35*-> '||TO_CHAR(ATAN(0.32))||' '||TO_CHAR(TAN(-43 *
3.14159)));

        -- Function  CEIL() --
        DBMS_OUTPUT.put_line('36*-> '||TO_CHAR(CEIL(17.134))||' '||TO_CHAR(CEIL(-
17.134)));

        -- Function  COS() SIN() --
        DBMS_OUTPUT.put_line('37*-> '||TO_CHAR(COS(90))||' '||TO_CHAR(SIN(60)));

        -- Function  EXP() --
        DBMS_OUTPUT.put_line('38*-> '||TO_CHAR(EXP(1))||' '||TO_CHAR(EXP(3.456)));

        -- Function  LN() LOG() --
        DBMS_OUTPUT.put_line('39*-> '||TO_CHAR(LN(100))||' '||TO_CHAR(LOG(5, 31))||'
'||TO_CHAR(LOG(2, 12)));

        -- Function  MOD() --
        DBMS_OUTPUT.put_line('40*-> '||TO_CHAR(MOD(34, 8))||' '||TO_CHAR(MOD(45, 7)));

        -- Function  POWER() --
        DBMS_OUTPUT.put_line('41*-> '||TO_CHAR(POWER(8, 4))||' '||TO_CHAR(POWER(65, -
7)));

        -- Function  ROUND() --
        DBMS_OUTPUT.put_line('42*-> '||TO_CHAR(ROUND(2.57))||' '||TO_CHAR(ROUND(3.678,
1))||' '||TO_CHAR(ROUND(14.8, -2)));

        -- Function  SIGN() --
        DBMS_OUTPUT.put_line('43*-> '||TO_CHAR(SIGN(-4))||' '||TO_CHAR(SIGN(0))||'
'||TO_CHAR(SIGN(6)));

        -- Function  SQRT() --
        DBMS_OUTPUT.put_line('44*-> '||TO_CHAR(SQRT(98))||' '||TO_CHAR(SQRT(9)));

        -- Function  TRUNC() --
        DBMS_OUTPUT.put_line('45*-> '||TO_CHAR(TRUNC(-123.456))||'
'||TO_CHAR(TRUNC(123.456,1))||' '||TO_CHAR(TRUNC(123.456, -1)));

  END;
/
```

Получаем после запуска:

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2
  3     -- Function  ATAN() TAN() --
  4     DBMS_OUTPUT.put_line('35*-> '||TO_CHAR(ATAN(0.32))||' '||TO_CHAR(TAN(-43 *
3.14159)));
  5
  6     -- Function  CEIL() --
  7     DBMS_OUTPUT.put_line('36*-> '||TO_CHAR(CEIL(17.134))||' '||TO_CHAR(CEIL(-
17.134)));
  8
  9     -- Function  COS() SIN() --
 10     DBMS_OUTPUT.put_line('37*-> '||TO_CHAR(COS(90))||' '||TO_CHAR(SIN(60)));
 11
 12     -- Function  EXP() --
```

```
 13     DBMS_OUTPUT.put_line('38*-> '||TO_CHAR(EXP(1))||' '||TO_CHAR(EXP(3.456)));
 14
 15     -- Function  LN() LOG() --
 16     DBMS_OUTPUT.put_line('39*-> '||TO_CHAR(LN(100))||' '||TO_CHAR(LOG(5, 31))||'
'||TO_CHAR(LOG(2, 12)));
 17
 18     -- Function  MOD() --
 19     DBMS_OUTPUT.put_line('40*-> '||TO_CHAR(MOD(34, 8))||' '||TO_CHAR(MOD(45, 7)));
 20
 21     -- Function  POWER() --
 22     DBMS_OUTPUT.put_line('41*-> '||TO_CHAR(POWER(8, 4))||' '||TO_CHAR(POWER(65, -
7)));
 23
 24     -- Function  ROUND() --
 25     DBMS_OUTPUT.put_line('42*-> '||TO_CHAR(ROUND(2.57))||' '||TO_CHAR(ROUND(3.678,
1))||' '||TO_CHAR(ROUND(14.8, -2)));
 26
 27     -- Function  SIGN() --
 28     DBMS_OUTPUT.put_line('43*-> '||TO_CHAR(SIGN(-4))||' '||TO_CHAR(SIGN(0))||'
'||TO_CHAR(SIGN(6)));
 29
 30     -- Function  SQRT() --
 31     DBMS_OUTPUT.put_line('44*-> '||TO_CHAR(SQRT(98))||' '||TO_CHAR(SQRT(9)));
 32
 33     -- Function  TRUNC() --
 34     DBMS_OUTPUT.put_line('45*-> '||TO_CHAR(TRUNC(-123.456))||'
'||TO_CHAR(TRUNC(123.456,1))||' '||TO_CHAR(TRUNC(123.456, -1)));
 35
 36  END;
 37  /
35*-> ,309702944542456199917380810392415670091 4
,000114104361604459415200698613184275542783
36*-> 18 -17
37*-> -,448073616129170152365477314399639507 42 -,3048106211022167056256494654784258407 8
38*-> 2,718281828459045235360287471352662497 76
31,689962805379164738838711100451206926 47
39*-> 4,605170185988091368035982909368728415 2 2,133656214977322644142031543095615100 56
3,584962500721156181453738943947816508 73
40*-> 2 3
41*-> 4096 ,000000000000203988884709418710246172033037848527959
42*-> 3 3,7 0
43*-> -1 0 1
44*-> 9,899494936611665341611821069467886549 99 3
45*-> -123 123,4 120

Процедура PL/SQL успешно завершена.
```

ATAN(x) TAN(x)

Возвращает арктангенс и тангенс **x** соответственно. Пример 35*.

CEIL(x)

Возвращает наименьшее целое число, большее или равное **x**. Пример 36*.

COS(x) SIN(x)

Возвращает косинус и синус **x** соответственно. Пример 37*.

EXP(x)

Возвращает **e** в степени **x**, где **e** = (2,718281828 основание натурального логарифма). Пример 38*.

LN(x)

Возвращает натуральный логарифм **x**, Пример 39*.

LOG(y, x)

Возвращает логарифм **y** по основанию **x**. Основание должно быть положительным числом, отличным от 0 или 1, а **y** может быть любым положительным числом. Пример 39*.

MOD(x, y)

Возвращает остаток от деления **x** нацело на **y**. Если **y** равно 0, то возвращается **x**. Пример 40*.

POWER(x, y)

Возвращает **x** в степени **y**. Основание **x** и порядок **y** могут быть не положительными целыми числами, но если **x** - отрицательное число, то **y** должен быть целым числом. Пример 41*.

ROUND(x, [,y])

Возвращает **x** округленное до **y** разрядов справа от десятичной точки. Значением по умолчанию для **y** является 0, при этом **x** округляется до ближайшего целого числа. Если **y** - отрицательное число, то округляются цифры слева от десятичной точки. **y** должен быть целым числом. Пример 42*.

SIGN(x)

Если **x < 0** возвращает -1, **x = 0** возвращает 0, **x > 0** возвращает 1. Пример 43*.

SQRT(x)

Возвращает квадратный корень **x**. Значение **x** не может быть отрицательным! Пример 44*.

TRUNC(x, [,y])

Возвращает **x** усеченное (не округленное !) до **y** десятичных разрядов. Значением по умолчанию, для **y** является 0, при этом **x** усекается до целого числа. Если **y** отрицательно, то усекаются цифры слева от десятичной точки. Пример 45*.

```
SET SERVEROUTPUT ON

BEGIN

        -- Function  ADD_MONTHS() --
        DBMS_OUTPUT.put_line('46*-> '||ADD_MONTHS('09-02-2000', 5));

        -- Function  ADD_MONTHS() --
        DBMS_OUTPUT.put_line('47*-> '||ADD_MONTHS('18-06-2001', 3));

        -- Function  ADD_MONTHS() --
        DBMS_OUTPUT.put_line('48*-> '||ADD_MONTHS('12-02-1981', 2));

        -- Function  LAST_DAY() --
        DBMS_OUTPUT.put_line('49*-> '||LAST_DAY('09-02-2000')||' '||LAST_DAY('19-02-
2001'));

        -- Function  LAST_DAY() --
        DBMS_OUTPUT.put_line('50*-> '||LAST_DAY('05-03-2002'));

        -- Function  MONTHS_BETWEEN() --
        DBMS_OUTPUT.put_line('51*-> '||MONTHS_BETWEEN('16-04-1973', '16-03-1997'));

        -- Function  MONTHS_BETWEEN() --
        DBMS_OUTPUT.put_line('52*-> '||MONTHS_BETWEEN('18-04-1972', '23-03-1961'));

        -- Function  NEW_TIME() --
        DBMS_OUTPUT.put_line( '53*-> '||TO_CHAR(NEW_TIME(TO_DATE('12-04-1971 12:00:00',
'DD-MM-YYYY HH24:MI:SS'),
                'PST', 'EST'), 'DD-MON-YYYY HH24:MI:SS')||': Pacific -> Eastern' );

        -- Function  NEXT_DAY() --
        DBMS_OUTPUT.put_line( '54*-> '||NEXT_DAY('07-04-2003', 'Понедельник'));

        -- Function  NEXT_DAY() --
        DBMS_OUTPUT.put_line( '55*-> '||NEXT_DAY('01-03-1998', 'Четверг'));

END;
/
```

После запуска в **SQL*Plus** получаем:

```
SQL> BEGIN
  2
  3     -- Function  ADD_MONTHS() --
  4     DBMS_OUTPUT.put_line('46*-> '||ADD_MONTHS('09-02-2000', 5));
  5
  6     -- Function  ADD_MONTHS() --
  7     DBMS_OUTPUT.put_line('47*-> '||ADD_MONTHS('18-06-2001', 3));
  8
  9     -- Function  ADD_MONTHS() --
 10     DBMS_OUTPUT.put_line('48*-> '||ADD_MONTHS('12-02-1981', 2));
 11
 12     -- Function  LAST_DAY() --
 13     DBMS_OUTPUT.put_line('49*-> '||LAST_DAY('09-02-2000')||' '||LAST_DAY('19-02-
2001'));
 14
 15     -- Function  LAST_DAY() --
 16     DBMS_OUTPUT.put_line('50*-> '||LAST_DAY('05-03-2002'));
 17
 18     -- Function  MONTHS_BETWEEN() --
 19     DBMS_OUTPUT.put_line('51*-> '||MONTHS_BETWEEN('16-04-1973', '16-03-1997'));
 20
 21     -- Function  MONTHS_BETWEEN() --
 22     DBMS_OUTPUT.put_line('52*-> '||MONTHS_BETWEEN('18-04-1972', '23-03-1961'));
 23
 24     -- Function  NEW_TIME() --
 25     DBMS_OUTPUT.put_line( '53*-> '||TO_CHAR(NEW_TIME(TO_DATE('12-04-1971 12:00:00',
'DD-MM-YYYY HH24:MI:SS'),
 26             'PST', 'EST'), 'DD-MON-YYYY HH24:MI:SS')||': Pacific -> Eastern' );
 27
 28     -- Function  NEXT_DAY() --
 29     DBMS_OUTPUT.put_line( '54*-> '||NEXT_DAY('07-04-2003', 'Понедельник'));
 30
 31     -- Function  NEXT_DAY() --
 32     DBMS_OUTPUT.put_line( '55*-> '||NEXT_DAY('01-03-1998', 'Четверг'));
 33
 34  END;
 35  /
46*-> 09.07.00
47*-> 18.09.01
48*-> 12.04.81
49*-> 29.02.00 28.02.01
50*-> 31.03.02
51*-> -287
52*-> 132,838709677419354838709677419354838 71
53*-> 12-АПР-1971 15:00:00: Pacific -> Eastern
54*-> 14.04.03
55*-> 05.03.98

Процедура PL/SQL успешно завершена.
```

Если у кого-то, данный блок вызвал ошибки ничего страшного нет, просто у вас скорее всего не выставлен язык системы, об этом мы уже говорили. А, так же формат даты в самой **Windows** установите **dd.MM.YYYY**. Я думаю, что ошибок не будет. Теперь, давайте разберемся с функциями. Данные функции, работают с датами и возвращают дату, за исключением функции **MONTHS_BETWEEN**, которая возвращает, значение типа **NUMBER**. Я считаю этот блок функций очень интересным и полезным для дальнейшего применения.

ADD_MONTHS(d,x)

Возвращает дату **d** плюс **x** месяцев. Значение **x** может быть любым целым числом. Если в месяце, полученном в результате, число дней меньше, чем в месяце **d** то, возвращается последний день месяца результата. Если, число дней не меньше то день месяца-результата и

день месяца **d** совпадают. Временные компоненты даты **d** и результата одинаковы. Пример 46*, 47*, 48*.

LAST_DAY(d)

Возвращает дату последнего дня того месяца, в который входит **d**. Эту функцию, можно применять для определения количества дней оставшихся в текущем месяце. Пример 49*, 50*.

MONTHS_BETWEEN(дата 1, дата 2)

Возвращает число месяцев между "дата 1" и "дата 2". Если дни в "дата 1" и "дата 2" или если обе даты являются последними днями своих месяцев. То, результат представляет собой целое число. В противном случае результат будет содержать дробную часть, по отношению, к 31-дневному месяцу. Пример 51*, 52*.

NEW_TIME(d, пояс 1, пояс 2)

Вот, долго думал, показывать эту функцию или нет, решил ладно, пусть будет. Хотя, я ей еще не пользовался, больно она, какая то... а в прочем решать вам! Итак:
Возвращает дату и время, часового пояса 2 для того момента когда, датой и временем, часового пояса 1 является **d**. Где пояс 1, пояс 2 это строки символов, для поясного времени Америки. В примере 'PST' и 'EST' это Тихоокеанское поясное время и восточное поясное время соответственно. Точно не знаю, есть ли эти строки для России, но вообще, в этом случае, я применял другие способы определения времени по поясам, как правило, в клиентских приложениях, например, применив **C++**. Но может, у кого есть другое мнение на сей счет! Кстати в примере показаны, функции преобразования, с которыми вы еще не сталкивались, по этому, пока можете сами разобраться. Если не совсем понятно, я о них еще расскажу! :) Пример 53*;

NEXT_DAY(d, строка_символов)

Возвращает дату первого дня, наступающего после даты **d** и обозначенного строкой символов. Строка символов указывает день недели на языке текущего сеанса. (Вот с 8.1.5 иногда бывают проблемы с передачей русских слов например, в сеанс **SQL*Plus**, но на удивление 9.0.2 они полностью отсутствуют!) Временной компонент возвращаемого значения тот же, что и временной компонент **d**. Регистр символов строки значения не имеет. Пример 54*, 55*.

Пишем наш маленький справочник далее, вот еще несколько функций работы с датами. Запишем такой блок:

```
SET SERVEROUTPUT ON

BEGIN

        -- Function  ROUND() --
        DBMS_OUTPUT.put_line( '56*-> '||ROUND(TO_DATE('15-05-1998'), 'MM'));

        -- Function  ROUND() --
        DBMS_OUTPUT.put_line( '57*-> '||ROUND(TO_DATE('15-05-1998'), 'DY'));

        -- Function  ROUND() --
        DBMS_OUTPUT.put_line( '58*-> '||ROUND(TO_DATE('15-05-1998'), 'WW'));

        -- Function  TRUNC() --
        DBMS_OUTPUT.put_line( '59*-> '||TO_CHAR(TRUNC(TO_DATE('15-05-1977 15:34:12',
                'DD-MM-YYYY HH24:MI:SS'), 'YEAR'), 'DD-MM-YYYY HH24:MI:SS'));

        -- Function  TRUNC() --
        DBMS_OUTPUT.put_line( '60*-> '||TO_CHAR(TRUNC(TO_DATE('12-04-1999 10:24:42',
                'DD-MM-YYYY HH24:MI:SS'), 'MM'), 'DD-MM-YYYY HH24:MI:SS'));

        -- Function  TRUNC() --
        DBMS_OUTPUT.put_line( '61*-> '||TO_CHAR(TRUNC(TO_DATE('02-09-2003 05:22:55',
                'DD-MM-YYYY HH24:MI:SS'), 'IW'), 'DD-MM-YYYY HH24:MI:SS'));
```

```
        -- Function  SYSDATE() --
        DBMS_OUTPUT.put_line( '62*-> Now Time is: '||TO_CHAR(SYSDATE,  'MONTH DD YYYY
HH24:MI:SS'));

        -- Function  SYSDATE() --
        DBMS_OUTPUT.put_line( '63*-> Now Time is: '||TO_CHAR(SYSDATE,  'DD-MM-YYYY
HH24:MI:SS'));

END;
/
```

После запуска в **SQL*Plus** получаем:

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2
  3     -- Function  ROUND() --
  4     DBMS_OUTPUT.put_line( '56*-> '||ROUND(TO_DATE('15-05-1998'), 'MM'));
  5
  6     -- Function  ROUND() --
  7     DBMS_OUTPUT.put_line( '57*-> '||ROUND(TO_DATE('15-05-1998'), 'DY'));
  8
  9     -- Function  ROUND() --
 10     DBMS_OUTPUT.put_line( '58*-> '||ROUND(TO_DATE('15-05-1998'), 'WW'));
 11
 12     -- Function  TRUNC() --
 13     DBMS_OUTPUT.put_line( '59*-> '||TO_CHAR(TRUNC(TO_DATE('15-05-1977 15:34:12',
 14            'DD-MM-YYYY HH24:MI:SS'), 'YEAR'), 'DD-MM-YYYY HH24:MI:SS'));
 15
 16     -- Function  TRUNC() --
 17     DBMS_OUTPUT.put_line( '60*-> '||TO_CHAR(TRUNC(TO_DATE('12-04-1999 10:24:42',
 18            'DD-MM-YYYY HH24:MI:SS'), 'MM'), 'DD-MM-YYYY HH24:MI:SS'));
 19
 20     -- Function  TRUNC() --
 21     DBMS_OUTPUT.put_line( '61*-> '||TO_CHAR(TRUNC(TO_DATE('02-09-2003 05:22:55',
 22            'DD-MM-YYYY HH24:MI:SS'), 'IW'), 'DD-MM-YYYY HH24:MI:SS'));
 23
 24     -- Function  SYSDATE() --
 25     DBMS_OUTPUT.put_line( '62*-> Now Time is: '||TO_CHAR(SYSDATE,  'MONTH DD YYYY
HH24:MI:SS'));
 26
 27     -- Function  SYSDATE() --
 28     DBMS_OUTPUT.put_line( '63*-> Now Time is: '||TO_CHAR(SYSDATE,  'DD-MM-YYYY
HH24:MI:SS'));
 29
 30  END;
 31  /
56*-> 01.05.98
57*-> 18.05.98
58*-> 14.05.98
59*-> 01-01-1977 00:00:00
60*-> 01-04-1999 00:00:00
61*-> 01-09-2003 00:00:00
62*-> Now Time is: НОЯБРЬ   15 2003 19:50:58
63*-> Now Time is: 15-11-2003 19:50:58

Процедура PL/SQL успешно завершена.
```

Итак, начинаем по немногу разбираться.

ROUND(d, [, формат])

Округляет дату **d**, до единицы указанной форматом. Формат соответствует следующей таблице (она же действует и для функции **TRUNC**):

Формат	Единица округления или усечения
CC, SCC	Век
SYYYY, YYYY, YEAR, SYEAR, YYY, YY, Y	Год (округляется до 1 июля)
IYYY, IYY, IY, I	Год ISO
Q	Квартал (округляется до шестнадцатого дня второго месяца квартала)
MONTH, MON, MM, RM	Месяц (округляется до шестнадцатого дня)
WW	То же день недели, что и первый день года
IW	То же день недели, что и первый день года ISO
W	То же день недели, что и первый день месяца
DDD, DD, J	День
Day, DY, D	Первый день недели
HH, HH12, HH24	Час
MI	Минута

Если "формат" не указан, применяется формат по умолчанию **'DD'**, который округляет **d** до ближайшего дня. Пример 56*, 57*, 58*.

TRUNC(d, [, формат])

Возвращает дату **d** усеченную до единицы, указанной "формат". Если "формат" не указан, применяется формат по умолчанию **'DD'**, который усекает **d** до ближайшего дня. Пример 59*, 60*, 61*.

SYSDATE

Вот, одна из наиболее полезных функций, которая есть в **PL/SQL**. Возвращает текущую дату и время в системе. Возвращаемы формат **DATE**. В примерах 62* и 63* хорошо видно как можно изменить выводимый тип для **SYSDATE**! Эти маски для преобразования, мы еще рассмотрим, хотя я уже почти все их показал! :)

Так, же в **PL/SQL** предусмотрены некоторые арифметические действия с датами, давайте рассмотрим основные из них:

Операция	Тип возвращаемого значения	Описание
d1 - d2	NUMBER (!)	Возвращается разница в днях между d1 и d2. Это значение является действительным числом, где дробная часть означает неполный день.
d1 + d2	-	ЗАПРЕЩЕНА (Сколько, я функций написал из за этого!!!)
d1 + n	DATE	К d1 добавляется n дней и возвращается результат, имеющий тип DATE. n может быть вещественным числом содержащим не полный день.
d1 - n	DATE	Из d1 вычитается n дней и возвращается результат, имеющий тип DATE. n может быть вещественным числом содержащим не полный день.

Давайте посмотрим несколько примеров:

```
DECLARE

D1 DATE;
D2 DATE;

BEGIN

        D1  := TO_DATE('12-10-2003 15:10:34', 'DD-MM-YYYY HH24:MI:SS');
        D2  := TO_DATE('11-03-2003 11:11:22', 'DD-MM-YYYY HH24:MI:SS');

        -- Function  SYSDATE() --
        DBMS_OUTPUT.put_line( '64*-> Today: '||TO_CHAR(SYSDATE,  'MONTH DD YYYY'));

        -- Function  SYSDATE() --
        DBMS_OUTPUT.put_line( '65*-> Tomorrow: '||TO_CHAR(SYSDATE - 1,  'MONTH DD
YYYY'));

        -- Function  SYSDATE() --
        DBMS_OUTPUT.put_line( '66*-> Return to The Future: '||TO_CHAR(SYSDATE + 1,
'MONTH DD YYYY'));

        -- Function  SYSDATE() --
        DBMS_OUTPUT.put_line( '67*-> Today: '||TO_CHAR(D1 - D2));


END;
/
```

После запуска в **SQL*Plus** получаем:

```
SQL> DECLARE
  2
  3  D1 DATE;
  4  D2 DATE;
  5
  6  BEGIN
  7
  8     D1  := TO_DATE('12-10-2003 15:10:34', 'DD-MM-YYYY HH24:MI:SS');
  9     D2  := TO_DATE('11-03-2003 11:11:22', 'DD-MM-YYYY HH24:MI:SS');
 10
 11     -- Function  SYSDATE() --
 12     DBMS_OUTPUT.put_line( '64*-> Today: '||TO_CHAR(SYSDATE,  'MONTH DD YYYY'));
 13
 14     -- Function  SYSDATE() --
 15     DBMS_OUTPUT.put_line( '65*-> Tomorrow: '||TO_CHAR(SYSDATE - 1,  'MONTH DD
YYYY'));
 16
 17     -- Function  SYSDATE() --
 18     DBMS_OUTPUT.put_line( '66*-> Return to The Future: '||TO_CHAR(SYSDATE + 1,
'MONTH DD YYYY'));
 19
 20     -- Function  SYSDATE() --
 21     DBMS_OUTPUT.put_line( '67*-> Today: '||TO_CHAR(D1 - D2));
 22
 23  END;
 24  /
64*-> Today: НОЯБРЬ   15 2003
65*-> Tomorrow: НОЯБРЬ   14 2003
66*-> Return to The Future: НОЯБРЬ   16 2003
67*-> Today: 215,166111111111111111111111111111111111

Процедура PL/SQL успешно завершена.
```

64* Показывает текущую дату, 65* возвращает нас в прошлое, (чем не машина времени!) 66* отправляет нас в будущее (Г. Уэллс просто отдыхает!) А, последний пример показывает разницу двух дат, в формате **NUMBER**! Вот так работают функции обработки дат! Пробуйте сами!!! :)