

## Using the TO\_NUMBER and TO\_DATE Functions

- Convert a character string to a number format using the TO\_NUMBER function:

```
TO_NUMBER(char[, 'format_model'])
```

- Convert a character string to a date format using the TO\_DATE function:

```
TO_DATE(char[, 'format_model'])
```

- These functions have an `fx` modifier. This modifier specifies the exact match for the character argument and date format model of a TO\_DATE function.

4-1

### Using the TO\_NUMBER and TO\_DATE Functions

You may want to convert a character string to either a number or a date. To accomplish this task, use the TO\_NUMBER or TO\_DATE functions. The format model that you select is based on the previously demonstrated format elements. The `fx` modifier specifies the exact match for the character argument and date format model of a TO\_DATE function:

Punctuation and quoted text in the character argument must exactly match (except for case) the corresponding parts of the format model.

The character argument cannot have extra blanks. Without `fx`, the Oracle server ignores extra blanks.

Numeric data in the character argument must have the same number of digits as the corresponding element in the format model. Without `fx`, the numbers in the character argument can omit leading zeros.

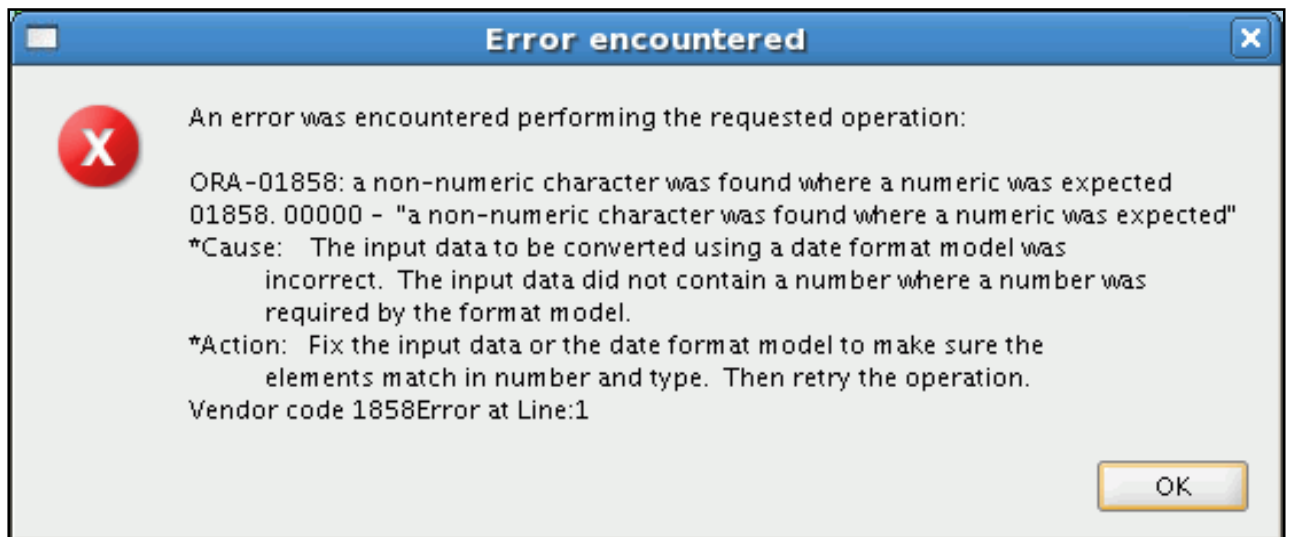
## Using the TO\_NUMBER and TO\_DATE Functions (continued)

### Example:

Display the name and hire date for all employees who started on May 24, 1999. There are two spaces after the month *May* and before the number 24 in the following example. Because the *fx* modifier is used, an exact match is required and the spaces after the word *May* are not recognized:

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May  24, 1999', 'fxMonth DD, YYYY');
```

The resulting error output looks like this:



To see the output, correct the query by deleting the extra space between 'May' and '24'.

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');
```

	LAST_NAME	HIRE_DATE
1	Grant	24-MAY-99

## Using the TO\_CHAR and TO\_DATE Function with the RR Date Format

To find employees hired before 1990, use the RR date format, which produces the same results whether the command is run in 1999 or now:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM employees
WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

	LAST_NAME	TO_CHAR(hire_date, 'DD-Mon-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

4-3

## Using the TO\_CHAR and TO\_DATE Function with the RR Date Format

To find employees who were hired before 1990, the RR format can be used.

Because the current year is greater than 1999, the RR format interprets the year portion of the date from 1950 to 1999.

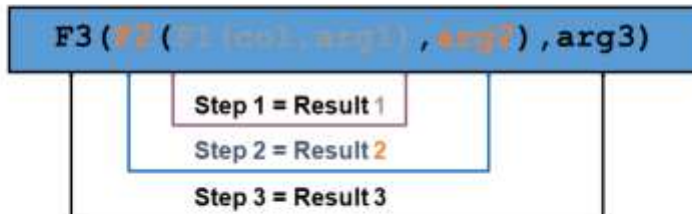
Alternatively, the following command, results in no rows being selected because the YY format interprets the year portion of the date in the current century (2090).

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM employees
WHERE TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

0 rows selected

## Nesting Functions

- Single-row functions can be nested to any level.
- Nested functions are evaluated from the deepest level to the least deep level.



4-4

### Nesting Functions

Single-row functions can be nested to any depth. Nested functions are evaluated from the innermost level to the outermost level. Some examples follow to show you the flexibility of these functions.

## Nesting Functions: Example 1

```
SELECT last name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME,1,8) , '_US'))  
FROM customers;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	OConnell	OCONNELL_US
2	Grant	GRANT_US
3	Whalen	WHALEN_US
4	Hartstein	HARTSTEI_US
5	Fay	FAY_US
6	Navier	NAVIER_US
7	Deer	DEER_US
8	Higgins	HIGGINS_US

■ ■ ■

4-5

### Nesting Functions (continued)

The example in the slide displays the last names of customers. The evaluation of the SQL statement involves three steps:

1. The inner function retrieves the first eight characters of the last name.  
`Result1 = SUBSTR (LAST_NAME, 1, 8)`
2. The outer function concatenates the result with \_US.  
`Result2 = CONCAT(Result1, '_US')`
3. The outermost function converts the results to uppercase.

The entire expression becomes the column heading because no column alias was given.

#### Example:

Display the date of the next Friday that is six months from the hire date. The resulting date should appear as Friday, August 13th, 1999. Order the results by hire date.

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS  
                     (hire_date, 6), 'FRIDAY'),  
         'fmDay, Month ddth, YYYY')  
      "Next 6 Month Review"
```

```
FROM employees  
ORDER BY hire_date;
```

## Nesting Functions: Example 2

```
SELECT TO_CHAR(ROUND((salary/7), 2), '99G999D99',  
            'NLS_NUMERIC_CHARACTERS = ','.')  
      "Formatted Salary"  
FROM employees;
```

	Formatted Salary
1	628,57
2	1.897,14
3	887,14
4	1.714,29
5	1.185,71
6	3.426,57

\*\*\*

4-6

### Nesting Functions (continued)

The example in the slide displays the salaries of employees divided by 7 and rounded to two decimals. The result is then formatted to display the salary in Danish notation. That is, comma is used for decimal point and a period for thousands.

First, the inner `ROUND` function is executed to round off the value of salary divided by 7 to two decimal places. The `TO_CHAR` function is then used to format the result of the `ROUND` function.

**Note:** D and G specified in the `TO_CHAR` function parameter are number format elements. D returns a decimal character in the specified position. G is used as a group separator.