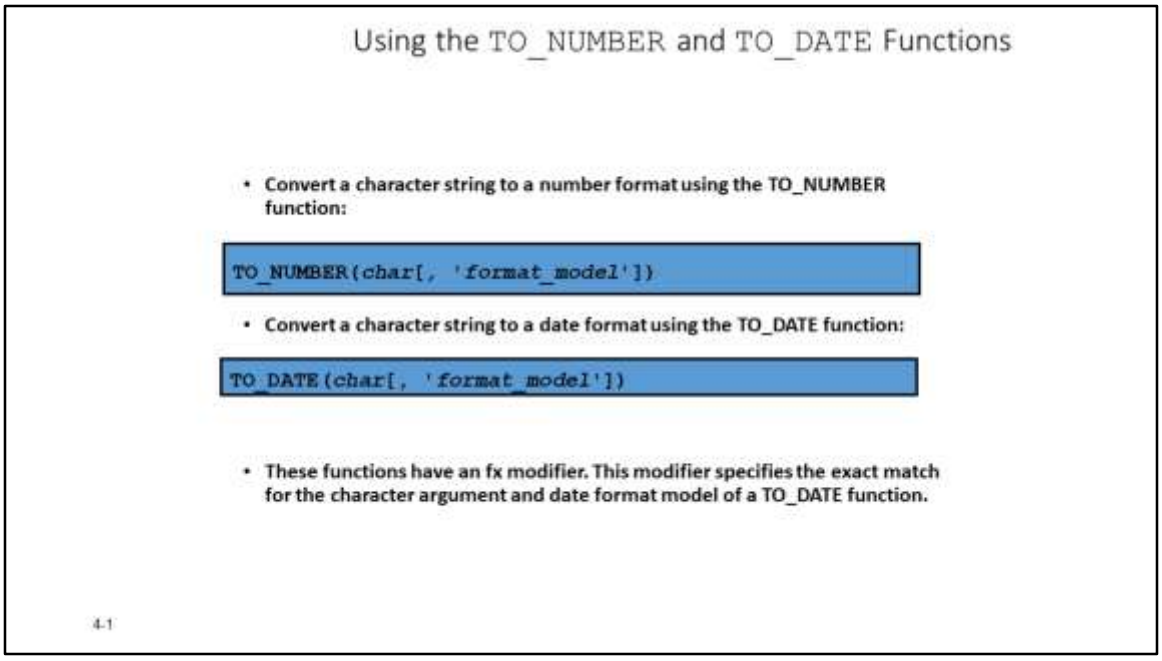


Using the TO_NUMBER and TO_DATE Functions

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.

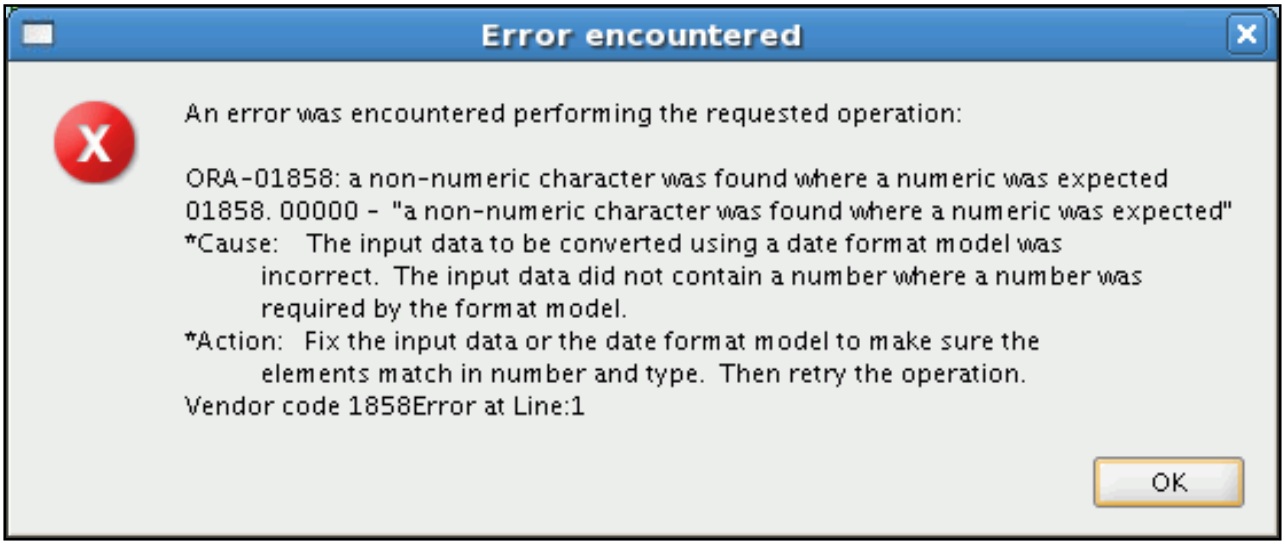
Using the TO_NUMBER and TO_DATE Functions (continued)

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');
```

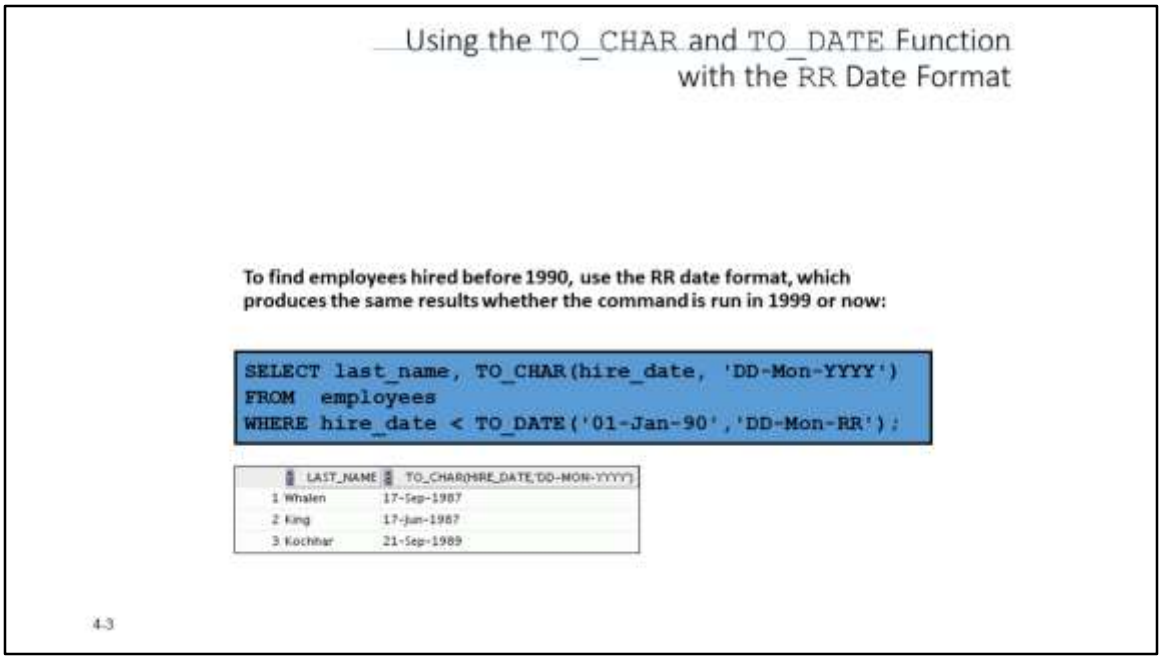
The resulting error output looks like this:



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



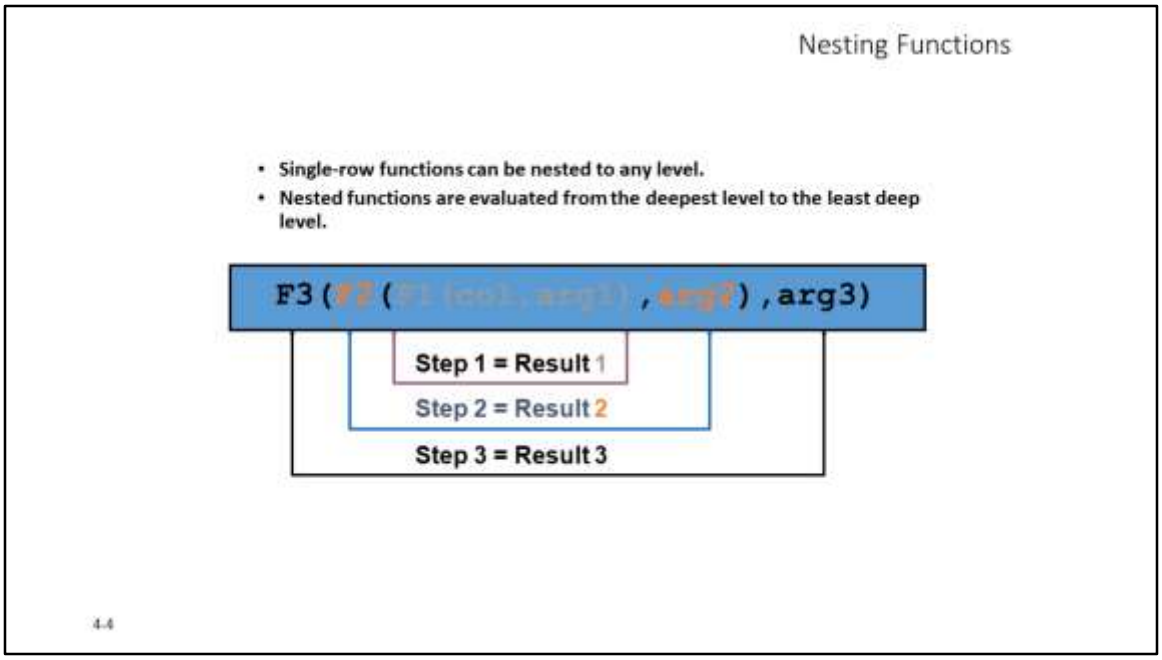
Using the TO_CHAR and TO_DATE Function with the RR Date Format

To find employees 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).

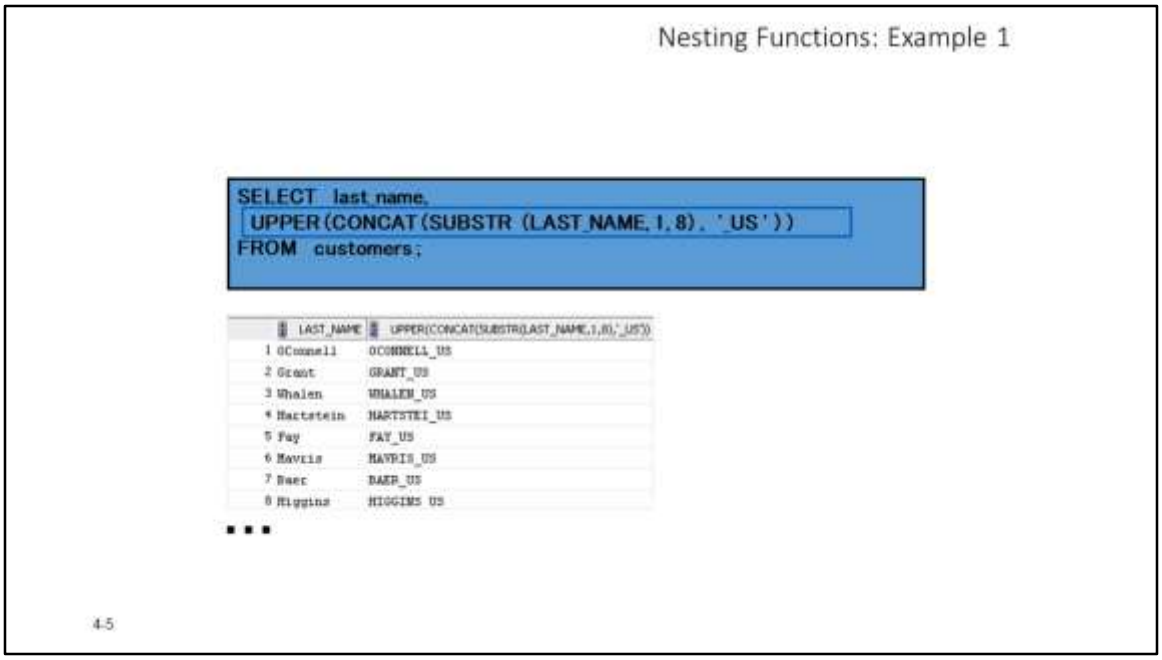
```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM   employees
WHERE  TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

```
0 rows selected
```



Nesting Functions

Single-row functions can be nested to any 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 (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.

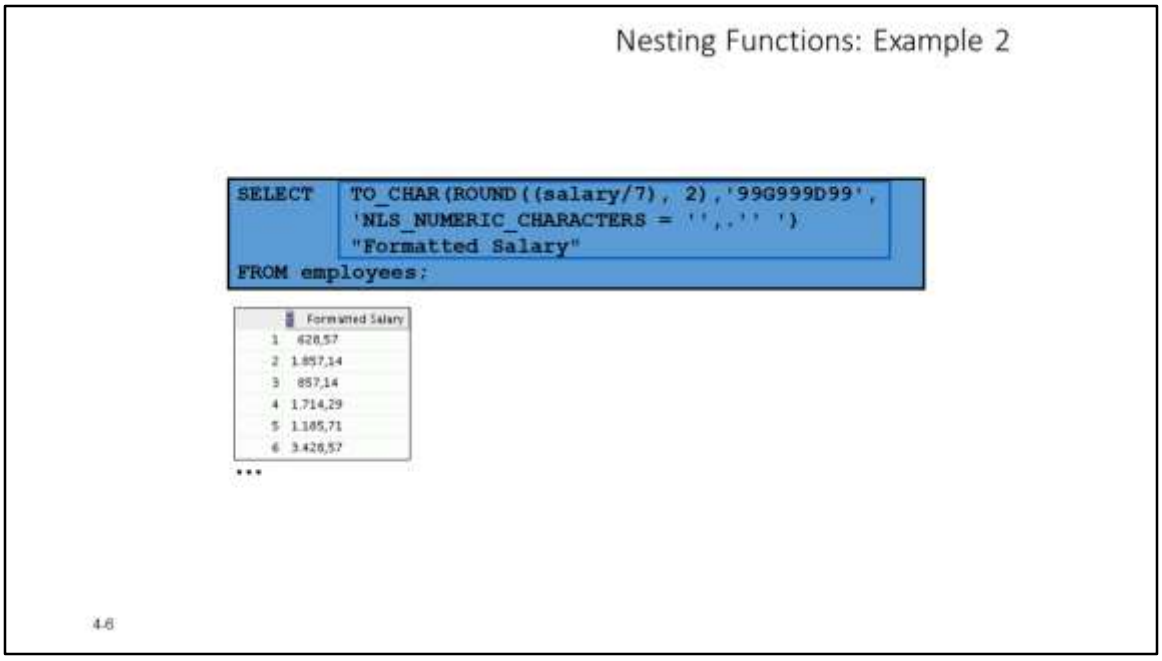
The entire expression becomes the column heading because no column alias was given.

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 (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.