5

Using Conversion Functions and Conditional Expressions

ORACLE
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe the various types of conversion functions that are available in SQL
- Use the `TO_CHAR`, `TO_NUMBER`, and `TO_DATE` conversion functions
- Apply conditional expressions in a `SELECT` statement

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

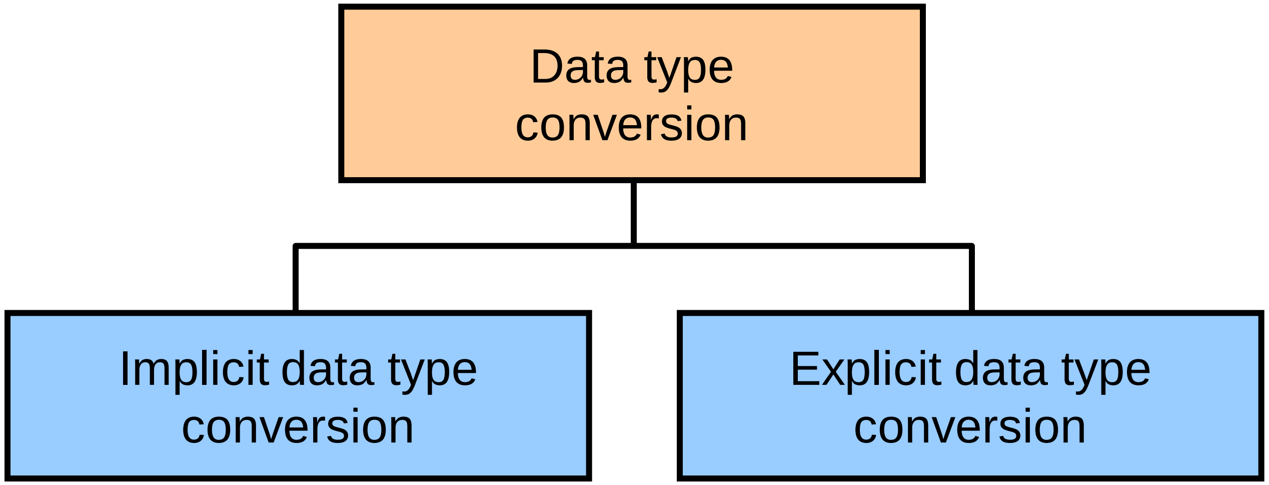
Lesson Agenda

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - Searched `CASE`
  - `DECODE`

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Conversion Functions



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Implicit Data Type Conversion

In expressions, the Oracle server can automatically convert the following:

From	To
`VARCHAR2 or CHAR`	`NUMBER`
`VARCHAR2 or CHAR`	`DATE`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

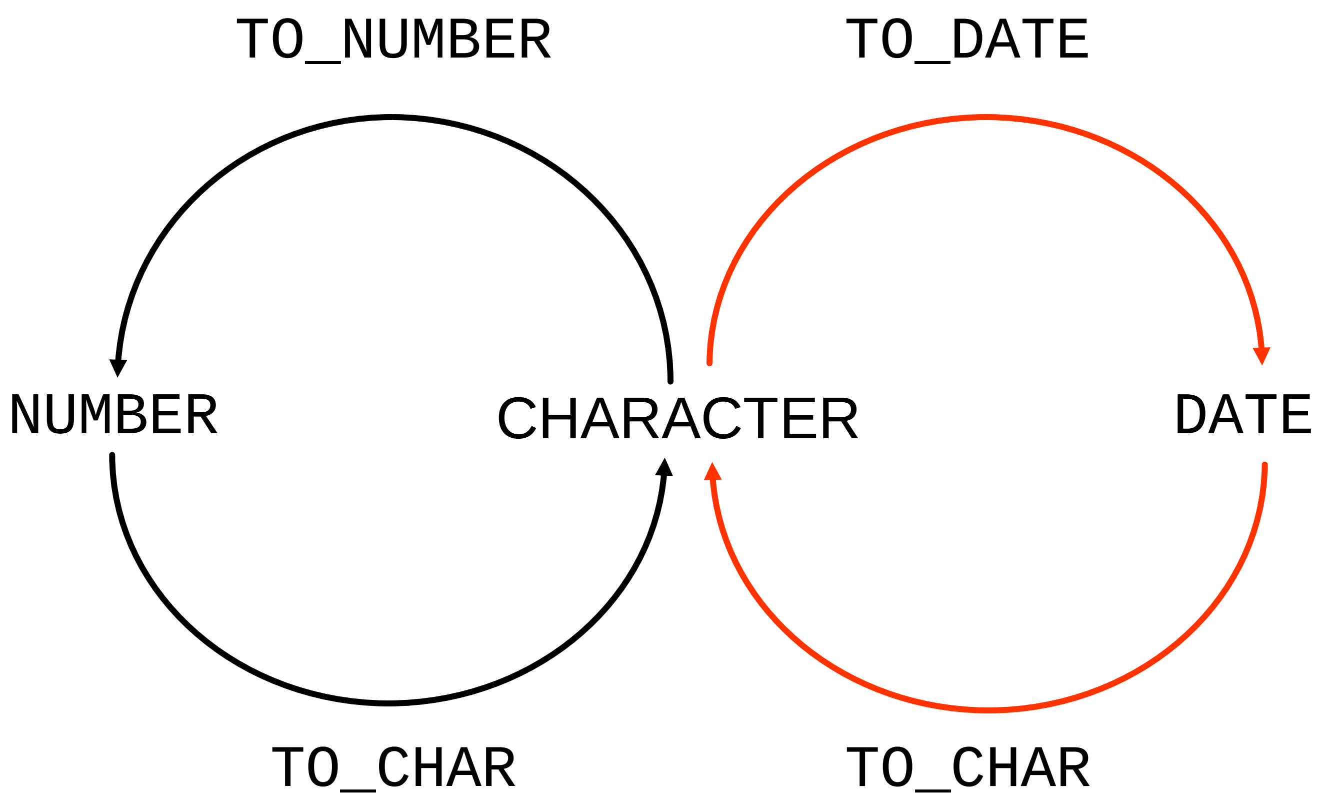
Implicit Data Type Conversion

For expression evaluation, the Oracle server can automatically convert the following:

From	To
`NUMBER`	`VARCHAR2 or CHAR`
`DATE`	`VARCHAR2 or CHAR`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Explicit Data Type Conversion



$$\begin{tikzcd}
\text{NUMBER} \arrow[r, bend right, "\text{TO\_CHAR}"'] & \text{CHARACTER} \arrow[l, bend right, "\text{TO\_NUMBER}"'] \arrow[r, bend left, "\text{TO\_DATE}"] & \text{DATE} \arrow[l, bend left, "\text{TO\_CHAR}"]
\end{tikzcd}$$

ORACLE
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - Searched `CASE`
  - `DECODE`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the TO_CHAR Function with Dates

```
TO_CHAR(date[,'format_model'])
```

The format model:

- Must be enclosed within single quotation marks
- Is case-sensitive
- Can include any valid date format element
- Has an `fm` element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Elements of the Date Format Model

Element	Result
`YYYY`	Full year in numbers
`YEAR`	Year spelled out (in English)
`MM`	Two-digit value for the month
`MONTH`	Full name of the month
`MON`	Three-letter abbreviation of the month
`DY`	Three-letter abbreviation of the day of the week
`DAY`	Full name of the day of the week
`DD`	Numeric day of the month

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Elements of the Date Format Model

- Time elements format the time portion of the date:

`HH24:MI:SS AM`	`15:45:32 PM`

- Add character strings by enclosing them within double quotation marks:

`DD "of" MONTH`	`12 of OCTOBER`

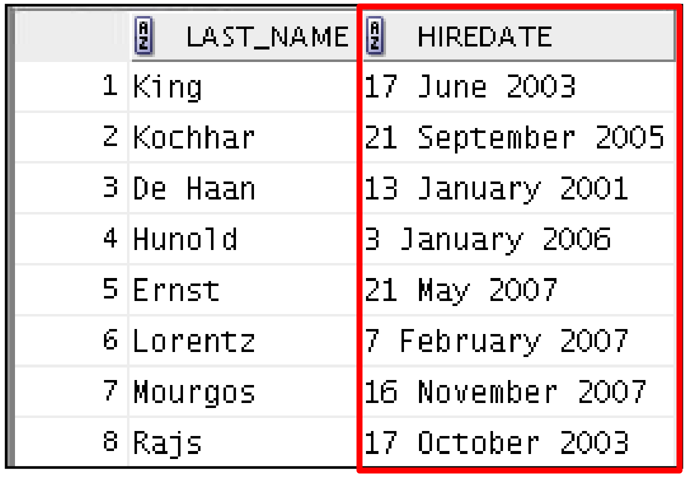
- Number suffixes spell out numbers:

`ddspth`	`fourteenth`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the TO_CHAR Function with Dates

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```



	LAST_NAME	HIREDATE
1	King	17 June 2003
2	Kochhar	21 September 2005
3	De Haan	13 January 2001
4	Hunold	3 January 2006
5	Ernst	21 May 2007
6	Lorentz	7 February 2007
7	Mourgos	16 November 2007
8	Rajs	17 October 2003

…

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the TO_CHAR Function with Numbers

```
TO_CHAR(number[, 'format_model'])
```

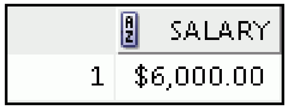
These are some of the format elements that you can use with the TO_CHAR function to display a number value as a character:

Element	Result
9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as a thousands indicator

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the TO_CHAR Function with Numbers

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```



```
     SALARY
1  $6,000.00
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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'])
```

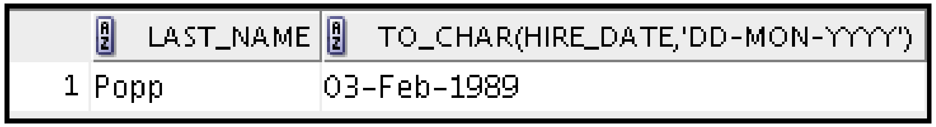
- These functions have an `fx` modifier. This modifier specifies the exact match for the character argument and date format model of a TO_DATE function.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using TO_CHAR and TO_DATE Functions 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	Popp	03-Feb-1989

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Implicit and explicit data type conversion
- TO_CHAR, TO_DATE, TO_NUMBER functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - CASE
  - Searched CASE
  - DECODE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

General Functions

The following functions work with any data type and pertain to using nulls:

- `NVL (expr1, expr2)`
- `NVL2 (expr1, expr2, expr3)`
- `NULLIF (expr1, expr2)`
- `COALESCE (expr1, expr2, ..., exprn)`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

NVL Function

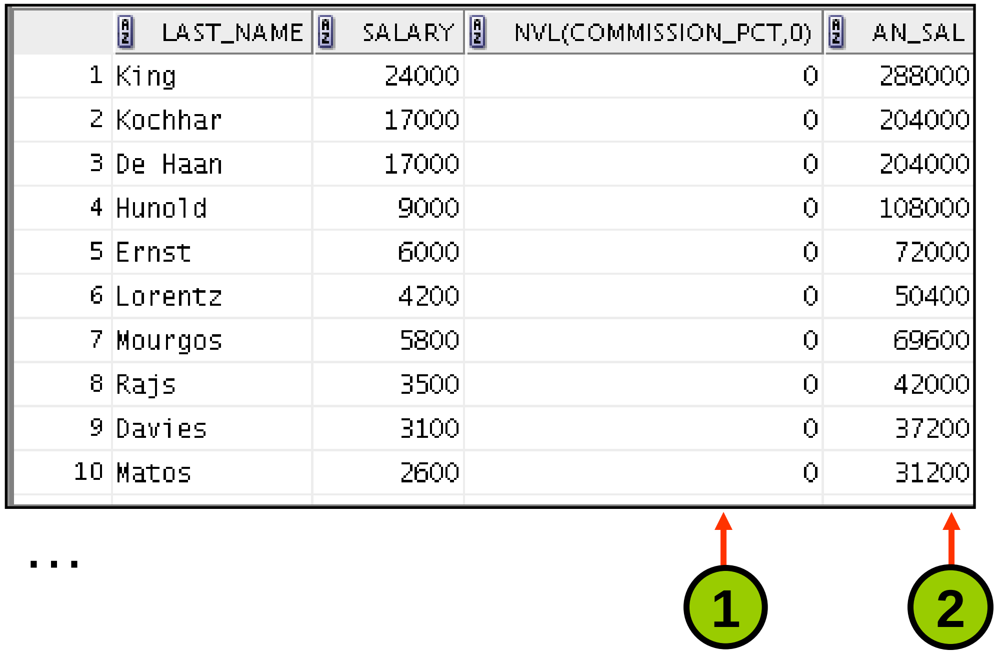
Converts a null value to an actual value:

- Data types that can be used are date, character, and number.
- Data types must match:
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the NVL Function

```
SELECT last_name, salary, NVL(commission_pct, 0),
   (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```



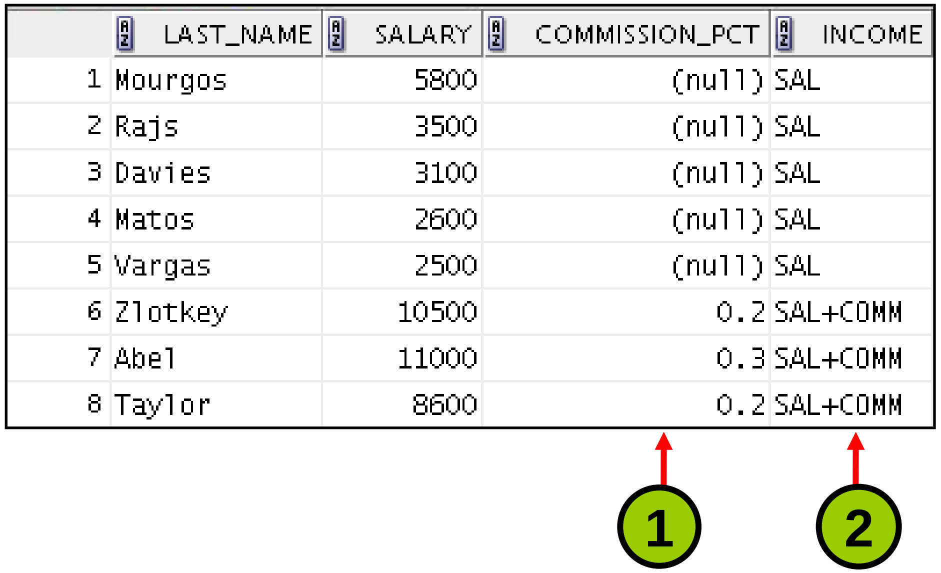
	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200

...

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the NVL2 Function

```
SELECT last_name,  salary, commission_pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

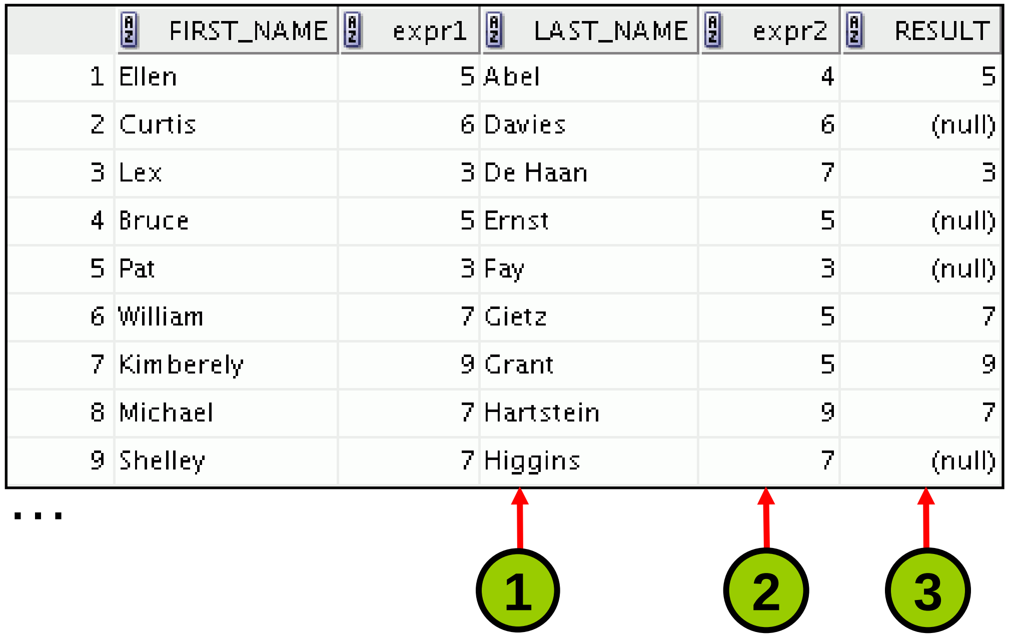


	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the NULLIF Function

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name,  LENGTH(last_name)  "expr2",
      NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM   employees;
```



	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9
8	Michael	7	Hartstein	9	7
9	Shelley	7	Higgins	7	(null)

...

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

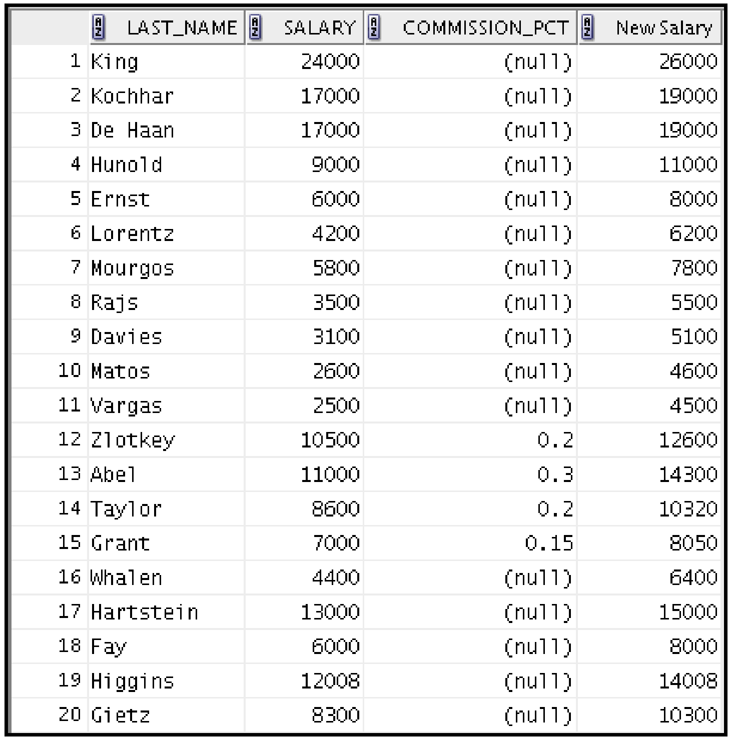
Using the `COALESCE` Function

- The advantage of the `COALESCE` function over the `NVL` function is that the `COALESCE` function can take multiple alternative values.
- If the first expression is not null, the `COALESCE` function returns that expression; otherwise, it does a `COALESCE` of the remaining expressions.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the COALESCE Function

```
SELECT last_name, salary, commission_pct,
COALESCE((salary+(commission_pct*salary)), salary+2000)"New Salary'
FROM   employees;
```



	LAST_NAME	SALARY	COMMISSION_PCT	New Salary
1	King	24000	(null)	26000
2	Kochhar	17000	(null)	19000
3	De Haan	17000	(null)	19000
4	Hunold	9000	(null)	11000
5	Ernst	6000	(null)	8000
6	Lorentz	4200	(null)	6200
7	Mourgos	5800	(null)	7800
8	Rajs	3500	(null)	5500
9	Davies	3100	(null)	5100
10	Matos	2600	(null)	4600
11	Vargas	2500	(null)	4500
12	Zlotkey	10500	0.2	12600
13	Abel	11000	0.3	14300
14	Taylor	8600	0.2	10320
15	Grant	7000	0.15	8050
16	Whalen	4400	(null)	6400
17	Hartstein	13000	(null)	15000
18	Fay	6000	(null)	8000
19	Higgins	12008	(null)	14008
20	Gietz	8300	(null)	10300

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - Searched `CASE`
  - `DECODE`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Conditional Expressions

- Provide the use of the `IF-THEN-ELSE` logic within a SQL statement
- Use the following methods:
  - `CASE` expression
  - Searched `CASE` expression
  - `DECODE` function

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

CASE Expression

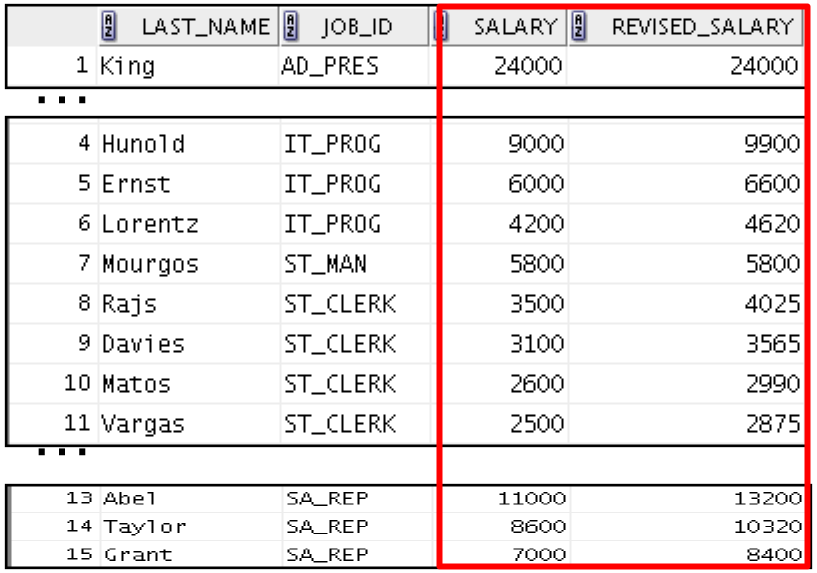
Facilitates conditional inquiries by doing the work of an `IF-THEN-ELSE` statement:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
         [WHEN comparison_expr2 THEN return_expr2
          WHEN comparison_exprn THEN return_exprn
          ELSE else_expr]
END
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the CASE Expression

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE      salary END      "REVISED_SALARY"
FROM   employees;
```



```
     LAST_NAME  JOB_ID     SALARY  REVISED_SALARY
   1 King       AD_PRES     24000           24000
 ...
   4 Hunold     IT_PROG      9000            9900
   5 Ernst      IT_PROG      6000            6600
   6 Lorentz    IT_PROG      4200            4620
   7 Mourgos    ST_MAN       5800            5800
   8 Rajs       ST_CLERK     3500            4025
   9 Davies     ST_CLERK     3100            3565
  10 Matos      ST_CLERK     2600            2990
  11 Vargas     ST_CLERK     2500            2875
 ...
  13 Abel       SA_REP      11000           13200
  14 Taylor     SA_REP       8600           10320
  15 Grant      SA_REP       7000            8400
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Searched CASE Expression

```
CASE
    WHEN condition1 THEN use_expression1
    WHEN condition2 THEN use_expression2
    WHEN condition3 THEN use_expression3
    ELSE default_use_expression
END
```

```
SELECT last_name,salary,
(CASE WHEN salary<5000 THEN 'Low'
      WHEN salary<10000 THEN 'Medium'
      WHEN salary<20000 THEN 'Good'
      ELSE 'Excellent'
END) qualified_salary
FROM employees;
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

DECODE Function

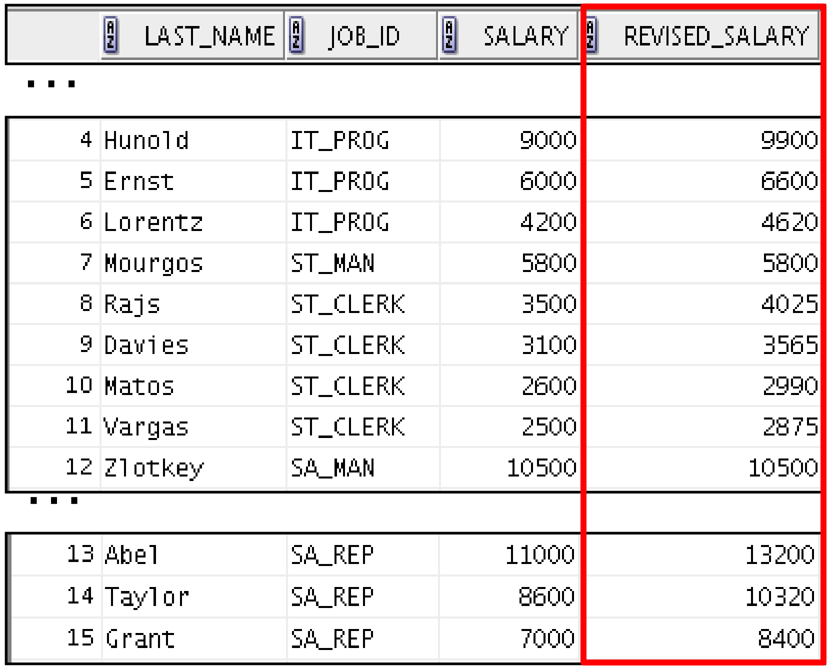
Facilitates conditional inquiries by doing the work of a CASE expression or an IF-THEN-ELSE statement:

```
DECODE(col|expression, search1, result1
                       [, search2, result2,...,]
                       [, default])
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the DECODE Function

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```



	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
4	Hunold	IT_PROG	9000	9900
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
10	Matos	ST_CLERK	2600	2990
11	Vargas	ST_CLERK	2500	2875
12	Zlotkey	SA_MAN	10500	10500
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
15	Grant	SA_REP	7000	8400

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using the DECODE Function

Display the applicable tax rate for each employee in department 80:

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                         0, 0.00,
                         1, 0.09,
                         2, 0.20,
                         3, 0.30,
                         4, 0.40,
                         5, 0.42,
                         6, 0.44,
                            0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Quiz

The `TO_NUMBER` function converts either character strings or date values to a number in the format specified by the optional format model.

a. True

b. False

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Alter date formats for display using functions
- Convert column data types using functions
- Use `NVL` functions
- Use `IF-THEN-ELSE` logic and other conditional expressions in a `SELECT` statement

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 5: Overview

This practice covers the following topics:

- Creating queries that use `TO_CHAR`, `TO_DATE`, and other `DATE` functions
- Creating queries that use conditional expressions such as `CASE`, searched `CASE`, and `DECODE`

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.