4

Using Conversion Functions and Conditional Expressions

Objectives

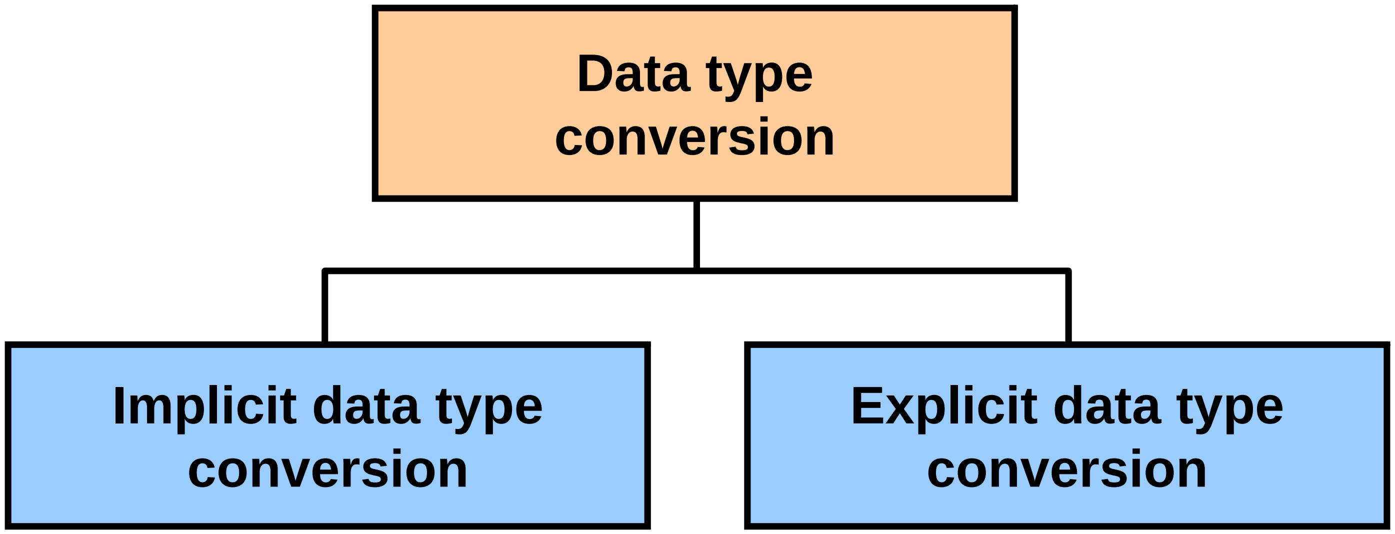
After completing this lesson, you should be able to do the following:

- 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

Lesson Agenda

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

Conversion Functions



Implicit Data Type Conversion

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

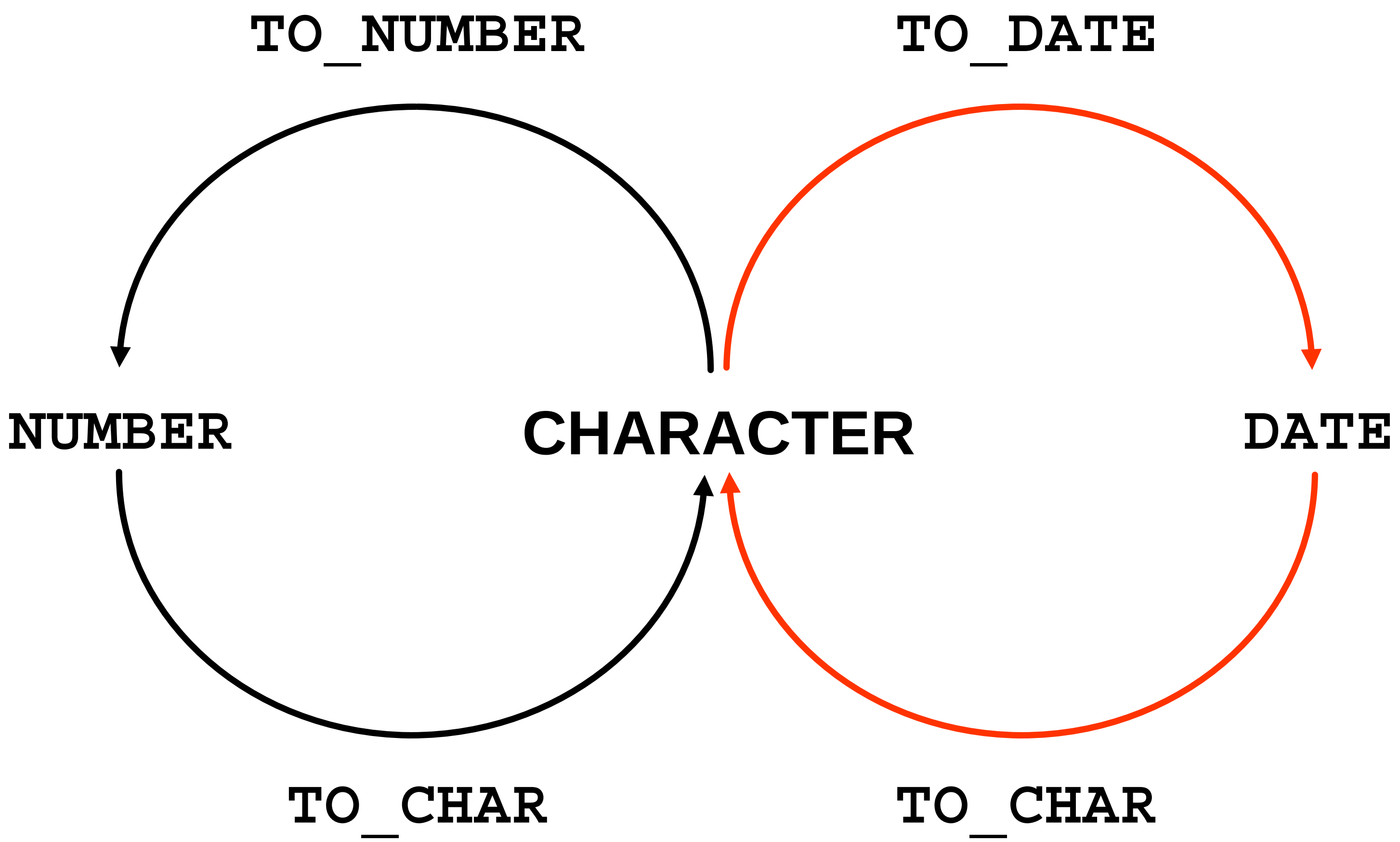
From	To
`VARCHAR2 or CHAR`	`NUMBER`
`VARCHAR2 or CHAR`	`DATE`

Implicit Data Type Conversion

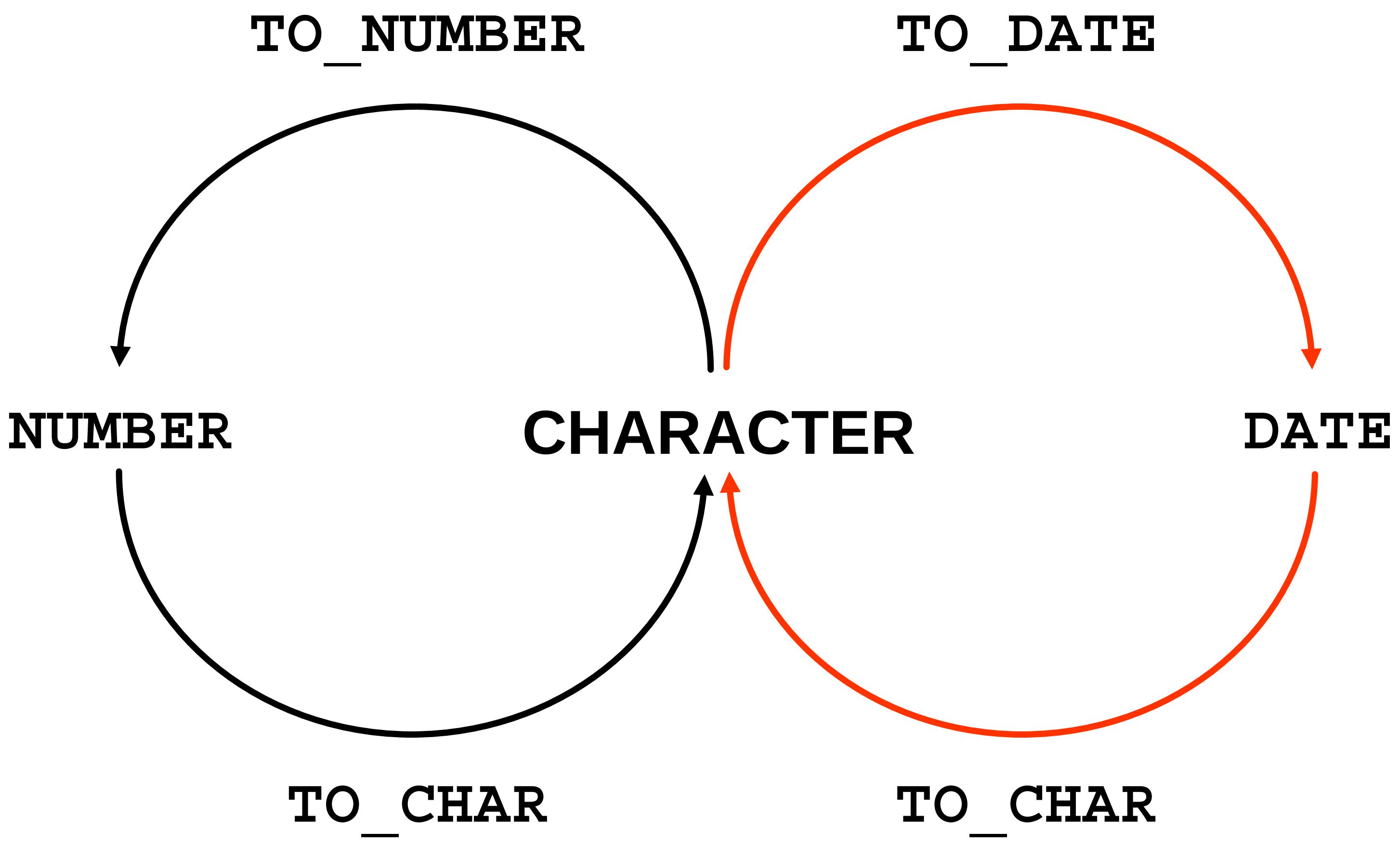
For expression evaluation, the Oracle server can automatically convert the following:

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

Explicit Data Type Conversion


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

Explicit Data Type Conversion



Lesson Agenda

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

Using the `TO_CHAR` Function with Dates

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

The format model:

- Must be enclosed with 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

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

ORACLE

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 with double quotation marks:

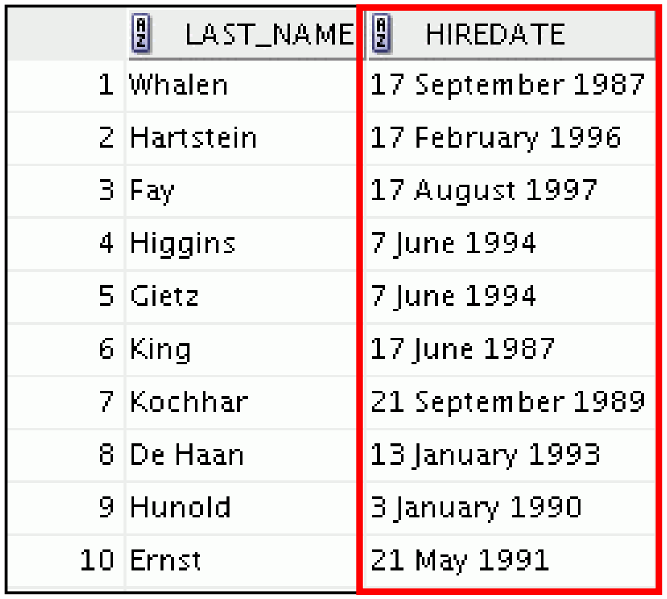
`DD "of" MONTH`	`12 of OCTOBER`

- Number suffixes spell out numbers:

`ddspth`	`fourteenth`

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	Whalen	17 September 1987
2	Hartstein	17 February 1996
3	Fay	17 August 1997
4	Higgins	7 June 1994
5	Gietz	7 June 1994
6	King	17 June 1987
7	Kochhar	21 September 1989
8	De Haan	13 January 1993
9	Hunold	3 January 1990
10	Ernst	21 May 1991

...

Using the `TO_CHAR` Function with Numbers

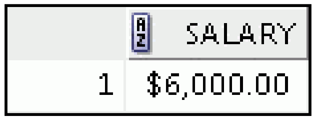
```
TO_CHAR(number, 'format_model')
```

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

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
```

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:

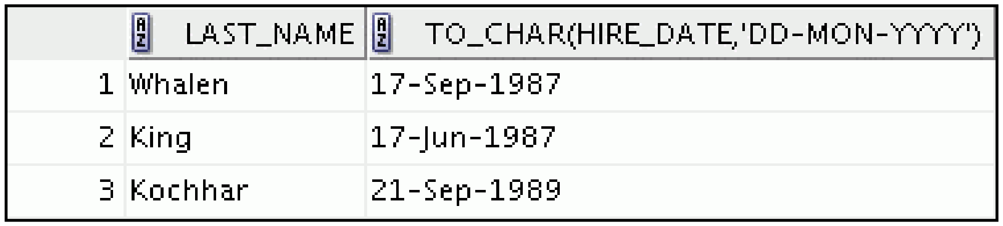
```
TO_DATE(char[, 'format_model'])
```

- 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.

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');
```



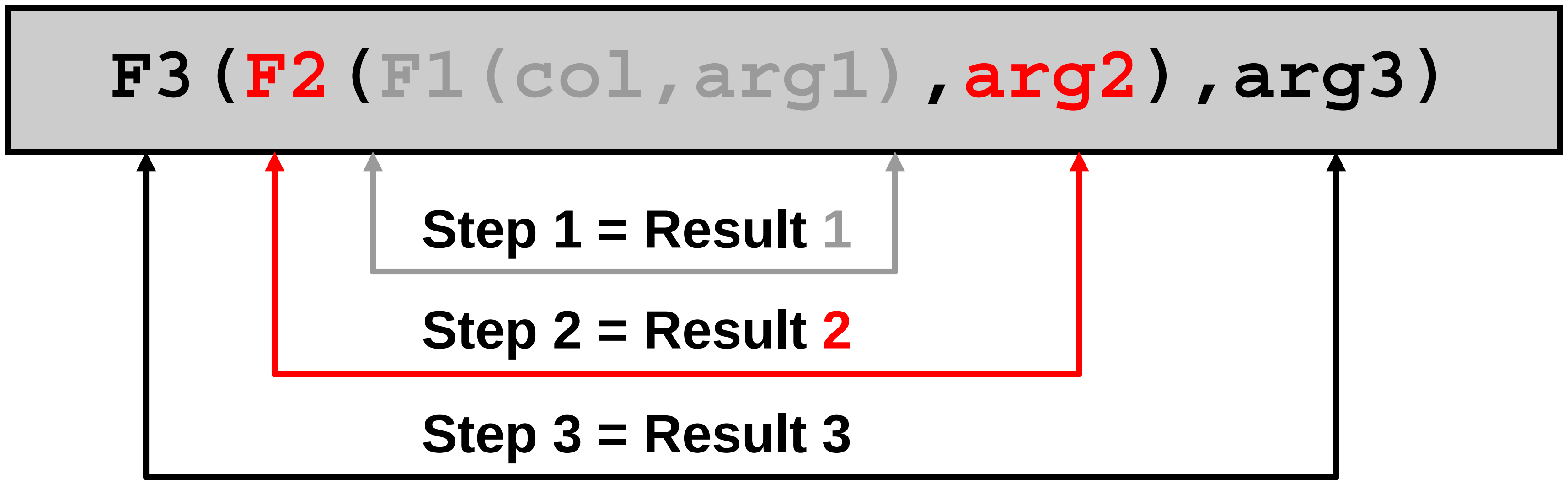
	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

Lesson Agenda

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

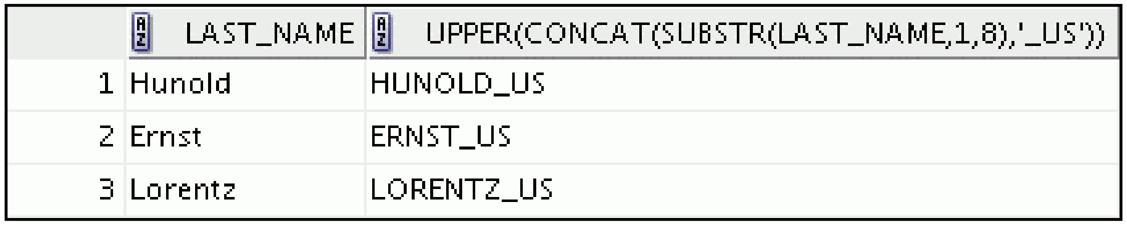
Nesting Functions

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



Nesting Functions: Example 1

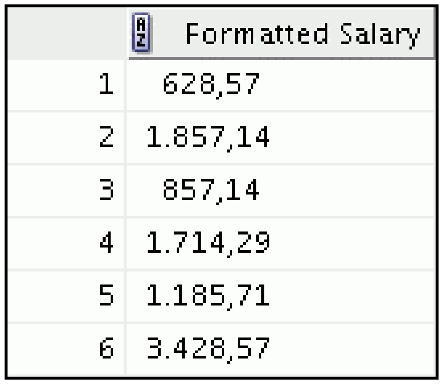
```
SELECT last_name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```



	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

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	1.857,14
3	857,14
4	1.714,29
5	1.185,71
6	3.428,57

…

Lesson Agenda

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

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)`

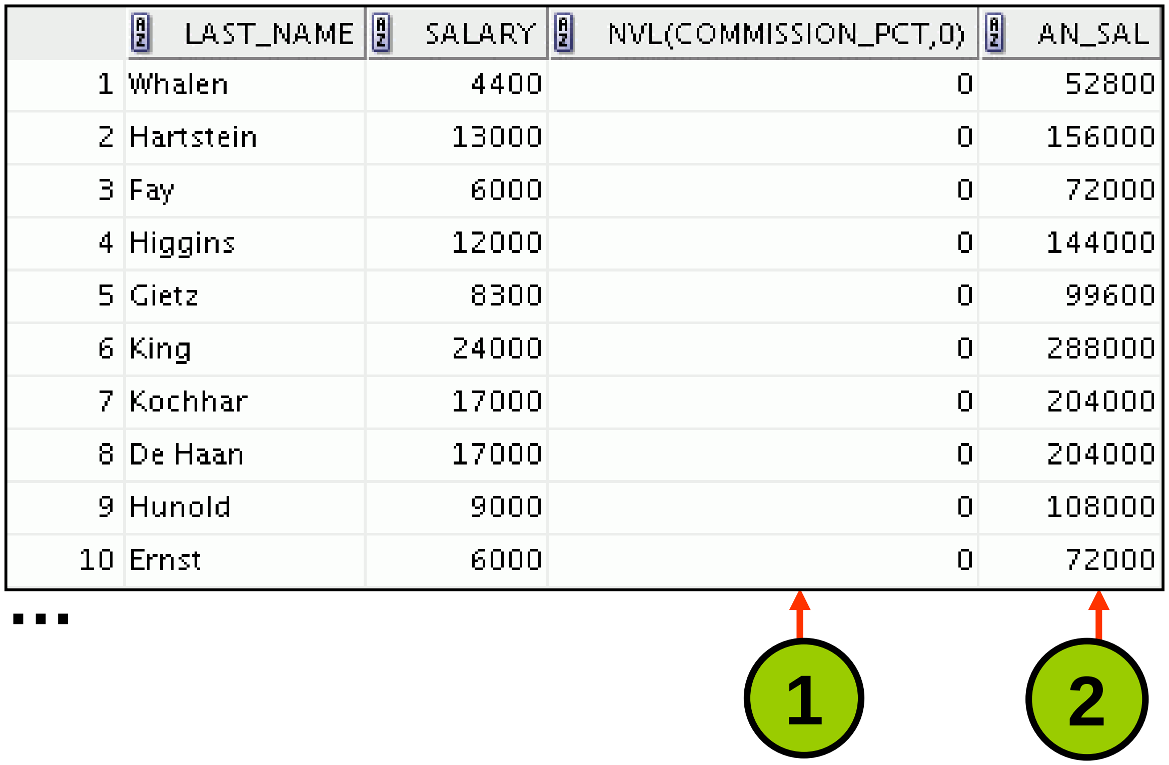
`NVL` Function

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')`

Using the NVL Function

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

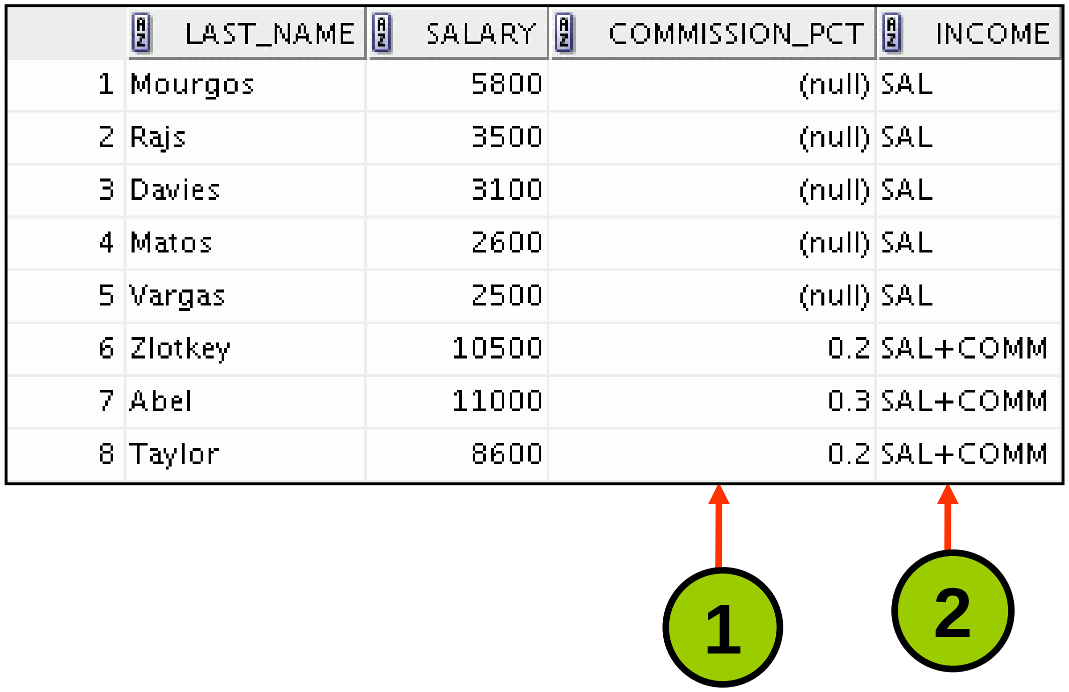


	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	Whalen	4400	0	52800
2	Hartstein	13000	0	156000
3	Fay	6000	0	72000
4	Higgins	12000	0	144000
5	Gietz	8300	0	99600
6	King	24000	0	288000
7	Kochhar	17000	0	204000
8	De Haan	17000	0	204000
9	Hunold	9000	0	108000
10	Ernst	6000	0	72000

...

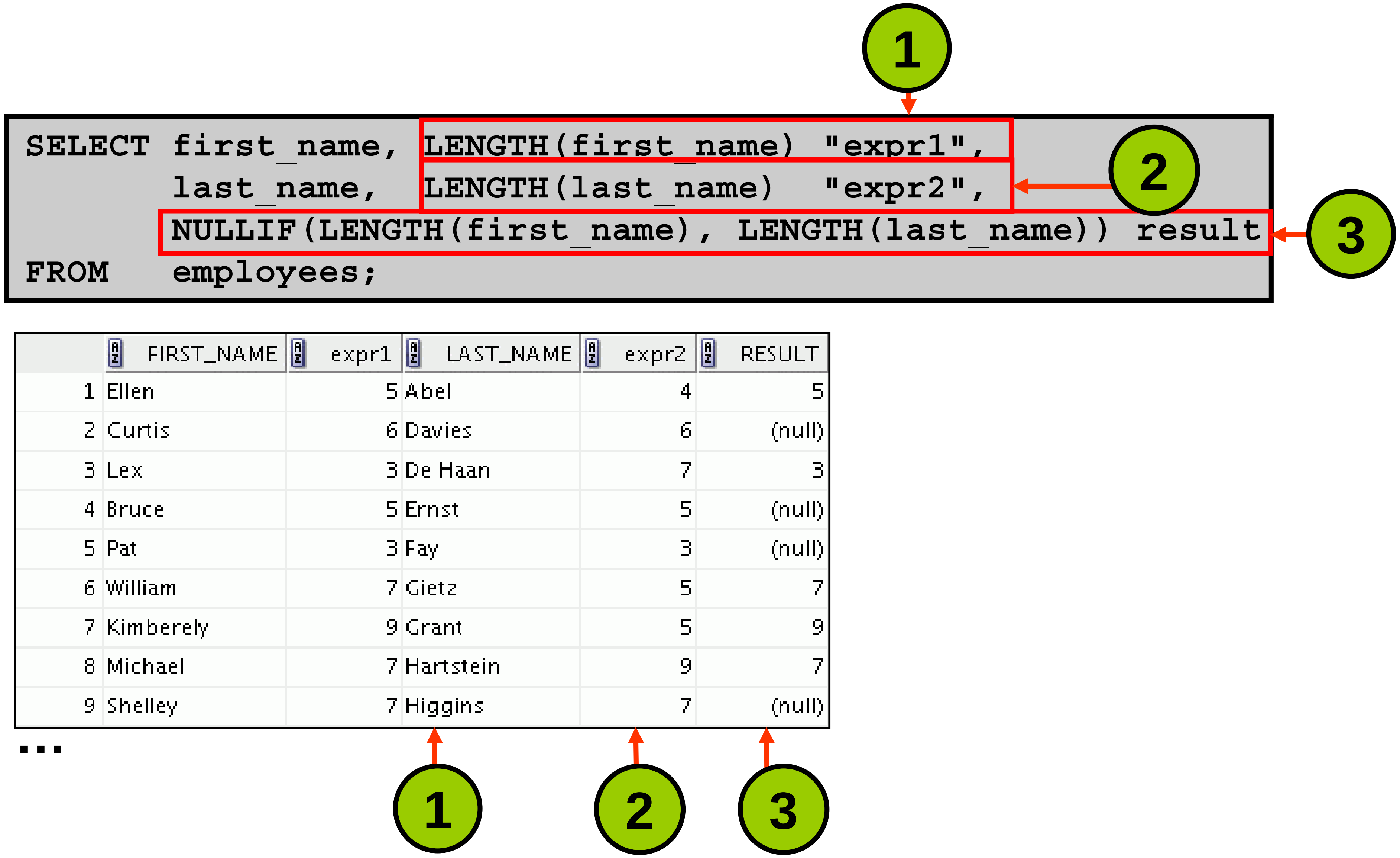
Using the NVL2 Function

```
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

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)

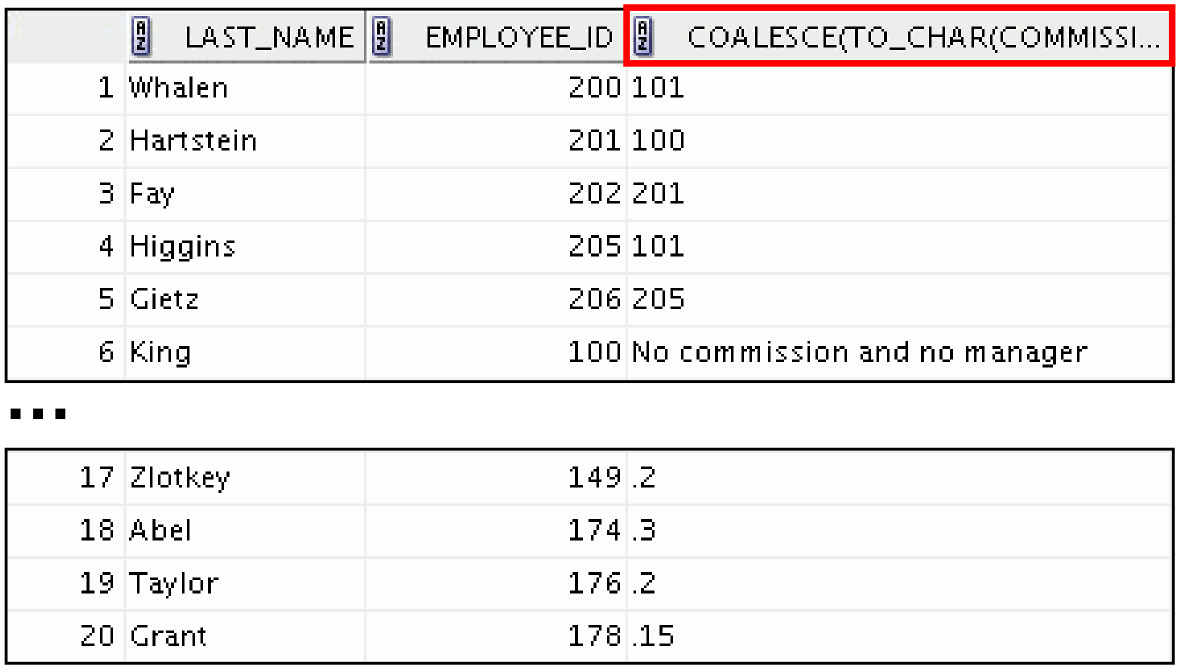
...

Using the `COALESCE` Function

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

Using the COALESCE Function

```
SELECT last_name, employee_id,
COALESCE(TO_CHAR(commission_pct),TO_CHAR(manager_id),
         'No commission and no manager')
FROM employees;
```



	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COMMISSI...
1	Whalen	200	101
2	Hartstein	201	100
3	Fay	202	201
4	Higgins	205	101
5	Gietz	206	205
6	King	100	No commission and no manager
...			
17	Zlotkey	149	.2
18	Abel	174	.3
19	Taylor	176	.2
20	Grant	178	.15

Lesson Agenda

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

Conditional Expressions

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

CASE Expression

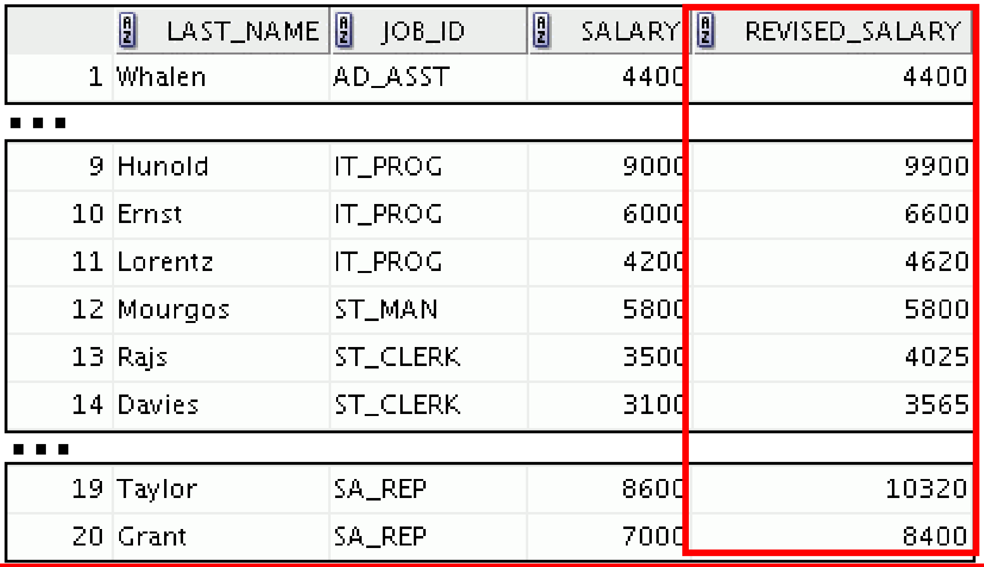
Facilitates conditional inquiries by doing the work of an `IF-THEN-ELSE` statement:

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

Using the `CASE` Expression

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

```
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	Whalen	AD_ASST	4400	4400
...				
9	Hunold	IT_PROG	9000	9900
10	Ernst	IT_PROG	6000	6600
11	Lorentz	IT_PROG	4200	4620
12	Mourgos	ST_MAN	5800	5800
13	Rajs	ST_CLERK	3500	4025
14	Davies	ST_CLERK	3100	3565
...				
19	Taylor	SA_REP	8600	10320
20	Grant	SA_REP	7000	8400

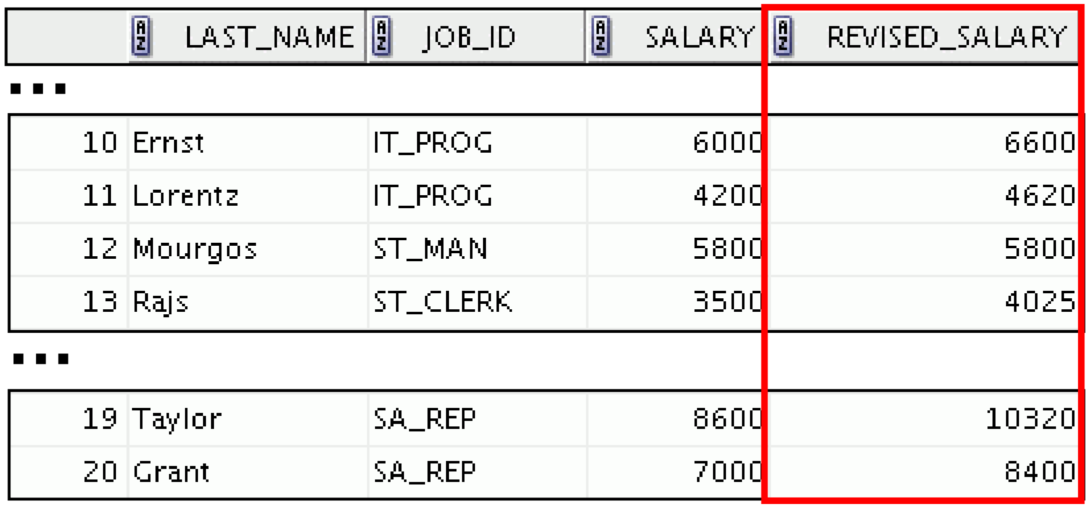
`DECODE` Function

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])
```

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
...				
10	Ernst	IT_PROG	6000	6600
11	Lorentz	IT_PROG	4200	4620
12	Mourgos	ST_MAN	5800	5800
13	Rajs	ST_CLERK	3500	4025
...				
19	Taylor	SA_REP	8600	10320
20	Grant	SA_REP	7000	8400

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;
```

Quiz

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

1. True
2. False

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

Practice 4: 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 `DECODE` and `CASE`